Worcester Polytechnic Institute
Digital WPI

Major Qualifying Projects (All Years) Major Qualifying Projects

March 2011

Pipeline ADC with a Nonlinear Gain Stage and Digital Correction

Anthony W. Crasso
Worcester Polytechnic Institute

K Zin Thet
Worcester Polytechnic Institute

Karen A. Anundson
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/mqp-all

Repository Citation
Crasso, A. W., Thet, K. Z., & Anundson, K. A. (2011). *Pipeline ADC with a Nonlinear Gain Stage and Digital Correction*. Retrieved from https://digitalcommons.wpi.edu/mqp-all/2417

This Unrestricted is brought to you for free and open access by the Major Qualifying Projects at Digital WPI. It has been accepted for inclusion in Major Qualifying Projects (All Years) by an authorized administrator of Digital WPI. For more information, please contact digitalwpi@wpi.edu.

Pipeline ADC with a Nonlinear Gain Stage and Digital Correction

A Major Qualifying Project

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

in

Electrical and Computer Engineering

by

Karen Anundson

Anthony Crasso

K Zin Thet

March 2011

APPROVED:

Professor John A. McNeill, Advisor

Abstract

The goal of this work was to design a pipeline analog to digital converter that can be calibrated and corrected in the digital domain. The scope of this work included the design, simulation and layout of major analog design blocks. The design uses an open loop gain stage to reduce power consumption, increase speed and relax small process size design requirements. These nonlinearities are corrected using a digital correction algorithm implemented in MATLAB.

Contents

1 Introduction **1**

2 Background **2**
- 2.1 Sampling . . . 2
  - 2.1.1 Oversampling Converter . . . 3
  - 2.1.2 Nyquist Converters . . . 3
- 2.2 Quantization . . . 4
- 2.3 Classification of ADCs . . . 4
  - 2.3.1 Low Speed, High Accuracy . . . 5
  - 2.3.2 Moderate Speed, Moderate Accuracy . . . 5
    - 2.3.2.1 SAR ADC . . . 5
    - 2.3.2.2 Cyclic ADC . . . 7
  - 2.3.3 High Speed . . . 7
    - 2.3.3.1 Flash . . . 7
    - 2.3.3.2 Pipeline . . . 8
- 2.4 Performance Metrics . . . 9
  - 2.4.1 Effective Number of Bits (ENOB) . . . 9
  - 2.4.2 Figure of Merit (FOM) . . . 9
  - 2.4.3 Differential Non-Linearity (DNL) . . . 10
  - 2.4.4 Integral Non-Linearity (INL) . . . 10
- 2.5 Calibration . . . 11
  - 2.5.1 Foreground Calibration . . . 11
  - 2.5.2 Background Calibration . . . 12
    - 2.5.2.1 Bootstrapped Digital Calibration . . . 12
    - 2.5.2.2 1.5-bit Stage ADC Architecture . . . 12
    - 2.5.2.3 Murmann's residue gain error correction . . . 12
    - 2.5.2.4 Split ADC Architecture . . . 13

3 System Level Design **16**
- 3.1 Design Specifications . . . 16
- 3.2 System Block Diagram . . . 17
  - 3.2.1 Stage Block Diagram . . . 18
  - 3.2.2 Timing . . . 20
  - 3.2.3 Stage Scaling . . . 20

4 Switched Capacitor Array **22**
- 4.1 Introduction . . . 22
- 4.2 Design and Simulation . . . 23
  - 4.2.1 Sample and Hold . . . 23
  - 4.2.2 Digital-to-Analog Converter . . . 26
  - 4.2.3 Reset Switching . . . 27
- 4.3 Layout . . . 28

5 Residue Amplifier 30
5.1 Introduction . . . 30
5.2 Design . . . 30
5.3 Fundamental Components of the Residue Amplifier . . . 31
5.3.1 Open Loop Differential Pair Amplifier . . . 31
5.3.1.1 Derivations of Resistive Load Values and Bias Current . . . 31
5.3.2 Replica Bias Circuit . . . 32
5.3.3 Source Follower . . . 33
5.4 Simulation and Results . . . 34
5.4.1 Gain . . . 34
5.4.2 Output Common Mode Voltage . . . 34
5.4.3 Settling Time . . . 35
5.4.4 Bandwidth . . . 36
5.4.5 Noise . . . 36
5.5 Complete Residue Amplifier . . . 39
5.6 Layout . . . 40

6 Sub-ADC 41
6.1 Comparator . . . 41
6.1.1 Preamplifier . . . 43
6.1.2 Analog Latch . . . 45
6.1.3 Complete Comparator . . . 48
6.2 Digital Latch and Logic . . . 49
6.3 Complete Sub-ADC Block . . . 51

7 2-bit Flash ADC 52
7.1 Comparators . . . 52
7.1.1 Preamplifier: ±1 . . . 52
7.1.2 Preamplifier: 0 . . . 54
7.1.3 Complete Comparators . . . 56
7.2 Complete 2-bit Flash . . . 57

8 Bias Current Source 58
8.1 Introduction . . . 58
8.2 Design . . . 58
8.3 Simulation . . . 59
8.4 Layout . . . 59

9 Timing Design 60
9.1 Introduction . . . 60
9.2 Design . . . 60

10 Results and Conclusions 64
10.1 Full Circuit Simulations . . . 64
10.2 Power . . . 67
10.3 Correction . . . 67
10.4 Signal-to-Noise Ratio and Figure of Merit . . . 70

11 Future Work 72
11.1 Reduction of Power Consumption . . . 72
11.2 Improving SNR and ENOB . . . 72
11.3 Continuing Correction and Calibration Implementation . . . 72

A MATLAB Correction Code 74

List of Figures

2.1 Analog-to-Digital Converter 2
2.2 Sampling 3
2.3 Quantization Error 4
2.4 ADC Architecture Comparison [1] 5
2.5 SAR Algorithm [3] 6
2.6 SAR ADC Topology [1] 6
2.7 Cyclic ADC Block Diagram[17] 7
2.8 Ruler Flash Analogy [1] 8
2.9 General Pipeline ADC Architecture[1] 9
2.10 Pipeline Stage with MDAC [1] 9
2.11 DNL 10
2.12 INL 11
2.13 Foreground Calibration Block Diagram 11
2.14 Difference Between Two Residue Characteristics 13
2.15 Error correction algorithms by using PDF 13
2.16 Split-ADC utilized in a pipeline ADC 14
2.17 Split ADC Architecture 14

3.1 System Block Diagram 17
3.2 Stage Block Diagram 18
3.3 Project Block Diagram 19
3.4 Detailed Switch System 20
3.5 System Timing 20
3.6 Scaling 21

4.1 System Block Diagram 22
4.2 Switch Capacitor Theory 22
4.3 Switch Capacitor Theory: Phase One 23
4.4 Switch Capacitor Theory: Phase Two 23
4.5 Top Plate Switching 24
4.6 Switch Capacitor Basic Timing 25
4.7 Differential Sample and Hold 25
4.8 Ideal Residue Plot 26
4.9 Switch Capacitor Array Switch Diagram 26
4.10 Switch Capacitor w/ DAC 27
4.11 Full Switch Capacitor w/ Reset Switching 28
4.12 Switch Capacitor Circuit Block 29
4.13 Switch Capacitor Array Layout 29

5.1 System Block Diagram with Highlighted Residue Amplifier 30
5.2 Residue Amplifier 31
5.3 Replica Bias Circuit 33
5.4 Transfer Function Plot of Differential Input to Differential Output 34

5.5 Transfer Function Plot of Differential Input to Each Output 35
5.6 Settling Time Plot 35
5.7 Bandwidth of the Residue Amplifier 36
5.8 Squared Output Noise Plot 37
5.9 Squared Output Noise Plot (Increased the Size of the Main Differential Pair) 37
5.10 Bandwidth Comparison of Design with W/L (blue) and 10W/10L (purple) 38
5.11 Bandwidth Comparison of Design with W/L (red) and 10W/10L (blue) 39
5.12 Symbol of Residue Amplifier 39
5.13 Layout of Residue Amplifier 40

6.1 Sub-ADC Block Diagram 41
6.2 Comparator 42
6.3 Comparator Block Diagram 42
6.4 DC Analysis 43
6.5 Preamplifier Circuit 44
6.6 Preamplifier Layout 45
6.7 Latch 46
6.8 Analog Latch Schematic 47
6.9 Analog Latch Transient Simulation 48
6.10 +1 Comparator Schematic 48
6.11 +1 Comparator Symbol 48
6.12 Combinational Logic 49
6.13 Logic with Digital Latches 50
6.14 Comparator Output Enable 50
6.15 Sub-ADC Block 51

7.1 +1 Preamplifier for 2-bit Flash 53
7.2 DC Analysis of +1 Preamplifier for 2-bit Flash 54
7.3 0 Preamplifier for 2-bit Flash 55
7.4 DC Analysis of 0 Preamplifier for 2-bit Flash 56
7.5 2-bit Flash ADC Comparator Symbols 56
7.6 2-bit Flash 57

8.1 Bias Current Source 58

9.1 Timing Diagram 60
9.2 Timing Schematic 61
9.3 Switch Capacitor Timing 61
9.4 Comparator Timing 62
9.5 Timing Blocks 63

10.1 Full Stage Schematic 65
10.2 Full System Residue 66
10.3 Stage One Residue Plot Input and Output 66
10.4 Inverse Look-Up Table 67
10.5 Non-linearity Comparison 69
10.6 Compared Output Code 70
10.7 Figure of Merit 71

List of Tables

2.1 Classification of ADCs [13] 4

3.1 Pipeline ADC Specifications 16

6.1 Combinational Logic for DAC Decisions 49

8.1 Bias Current Source Currents 59

Chapter 1

Introduction

Analog to digital converters are vital to many modern systems that require the integration of analog signals with digital systems. These applications can range from music recording to communications applications to medical instrumentation. [4] These converters are implemented using a variety of architectures, sizes and speeds. The demand for smaller, faster, lower power converters has led to the investigation of alternative ADC design techniques. As CMOS technologies improve and smaller process sizes lead to an increase in the implementation of digital signal processing, the potential for digital correction and calibration of ADCs has emerged. [12]

The goal of this work was to design a pipeline analog to digital converter that can be calibrated and corrected in the digital domain. The ADC was designed for use in a split ADC architecture for calibration. This design builds on previous work that uses a split ADC architecture on a design with non linear gain and extends this concept into the pipeline architecture.

The design features a nonlinear open loop gain stage. Using this open loop stage reduces power consumption and increases speed over its closed loop counterpart. It also significantly simplifies the design of the amplifier. The nonlinearities in the gain stage can be corrected using a digital calibration algorithm that has low power and space requirements. This transfers the design complexity from the analog domain to the digital domain, which is extremely desirable in smaller process sizes.

The scope of this work included the design, simulation and layout of major design blocks. The functional blocks were also combined to simulate full conversions. The results from these simulations were corrected using a digital correction algorithm adapted from previous work on a cyclic ADC. The results of this work are competitive with other pipeline ADC designs described in literature.

Chapter 2

Background

Analog-to-digital converters (ADCs) provide a link between the analog signals of the real world and the world of digital signal and data processing. Figure 2.1 shows the basic concept of an analog to digital converter: a continuous analog signal input is converted to a discrete digital signal at the output. This digital output can then be processed by a digital system such as a processor or an FPGA.

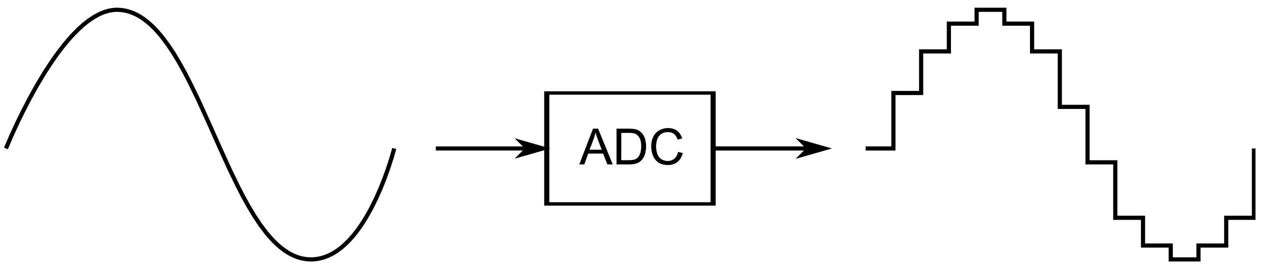


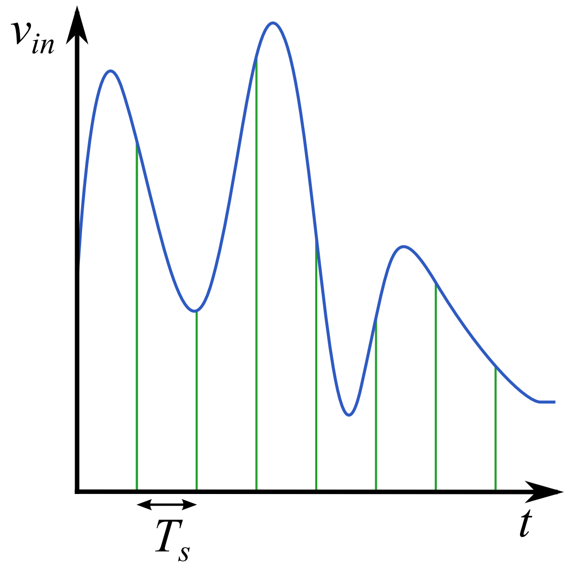
Figure 2.1: Analog-to-Digital Converter

The rapid growth and improvement of digital processing systems has led to more processing being implemented in the digital domain. Decreasing process sizes mean increased numbers of logic gates in a given space. The computational power of a digital system increases with the number of logic gates. Digital processing can often offer advantages in design flexibility. An FPGA, for example, allows for digital hardware designs to be reconfigured to suit changing system needs. In order to take advantage of these digital processing systems, however, real world analog signals must be converted into digital signals. As its name implies, an analog to digital converter fills this need.

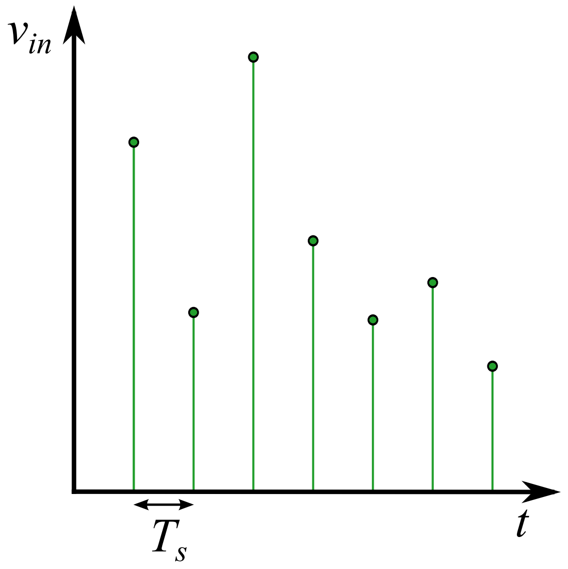
ADCs can be designed with a variety of architectures depending on the requirements for the device. Some of these architectures also include calibration methods to improve the ADC's performance. This section will introduce ADC concepts and architecture types and calibration.

2.1 Sampling

One of the fundamental parts of an analog to digital converter is a sampling component. In order for a continuous time analog signal to be converted to a discrete time signal, the analog signal must be sampled in time. Figure 2.2a shows a signal v_{in} being sampled every time T_s. Equivalently, the signal is being sampled at a frequency f_s. Ideally, the sampled input will be a series of impulses, shown in Figure 2.2b, with time spacing T_s and an amplitude determined by the value of the input signal at time nT_s, where n is an integer.



(a) Continuous Signal



(b) Discrete Time Sampled Signal

Figure 2.2: Sampling

Choosing a sampling frequency to ensure that the sampled signal contains sufficient information about the original signal and prevents aliasing can be done based on the Nyquist-Shannon sampling theorem. That is, if the sampling frequency

$$f_s > 2B_{signal} \tag{2.1}$$

then the signal can be fully recovered. This holds as long as the samples are not restricted to discrete y values as they are in a digital signal. The discrete behavior of the y values introduces errors due to quantization [5].

Analog-to-digital converters can be categorized into two major categories based on their sampling frequencies: oversampling and Nyquist converters.

2.1.1 Oversampling Converter

Oversampling converters are characterized by a sampling frequency much higher than the Nyquist rate. This high sampling rate causes larger spacing in the signal spectrum, ideally preventing the overlap of samples in the spectrum that leads to aliasing effects. These converters are typically used when high accuracy is required and a reduction in the effects of aliasing is desired, such as in bandlimited signals like music. The design trade-off for the accuracy is a lower throughput. These converters also require a large number of samples to perform a single conversion. [4]

2.1.2 Nyquist Converters

Nyquist converters can process signals up to one half of the sampling frequency. This is in accordance with the Nyquist theorem that the sampling frequency must be at least twice the bandwidth of the signal in order to recover the information from the original signal. That is

$$f_s = 2 * Bandwidth_{inputsignal} \tag{2.2}$$

These converters have higher throughput than oversampling converters. The trade-off made for this speed is a reduced accuracy. Some Nyquist converters are high speed with what is considered to be low to medium accuracy, such as flash or pipeline ADCs. Other Nyquist converters fall into the middle range for both speed and accuracy, such as successive approximation converters (SAR) and cyclic converters. These converters tend to be a good compromise between slow oversampling converters and less accurate options such as flash converters. [17]

2.2 Quantization

Quantization is also necessary for analog-to-digital converters. Quantization is the process of assigning certain ranges of values from a continuous signal range to discrete values. This assignment creates quantization errors. A quantization error is the difference between the quantized value and the original signal. In Figure 2.3a, the original signal v_{in} is shown in blue. If a sample of this signal is taken at time T_1, it would be quantized to n_2, as shown in Figure 2.3b. The difference between the sample of v_{in} and its quantized value n_2 is indicated by the black bar.

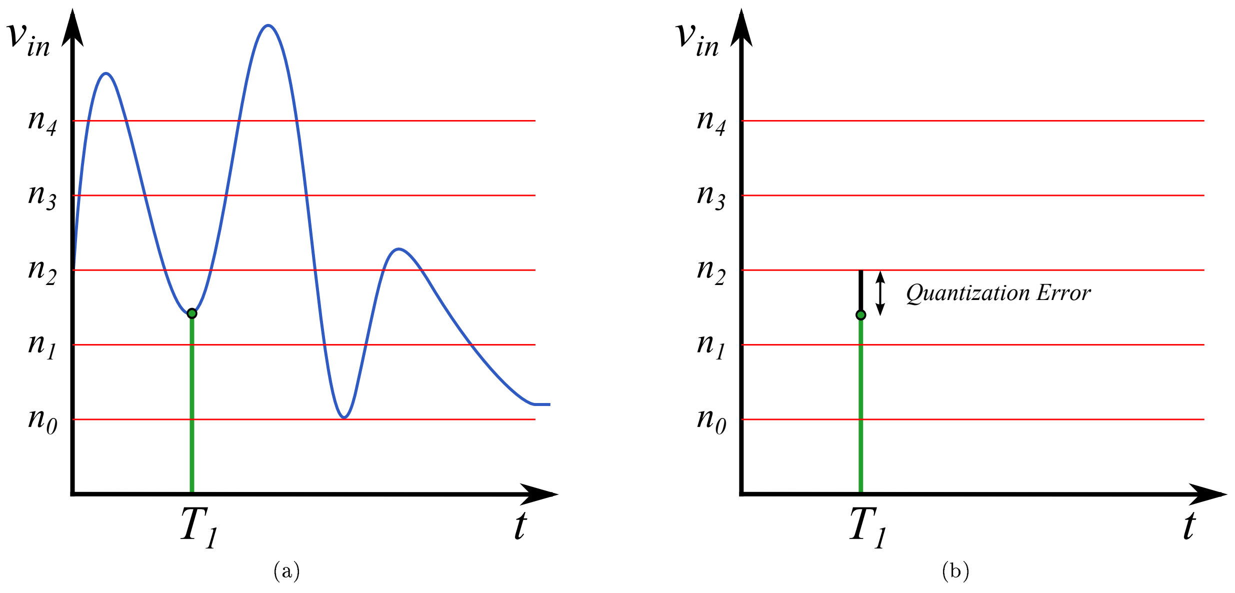


Figure 2.3: Quantization Error

Quantization errors are directly related to the resolution of the ADC. An ADC that needs an accuracy within a very small margin of error is going to need more quantization levels. More levels require a larger number of digital bits to encode all the information. Higher resolution often comes at the cost of converter speed, so converters need to be optimized for required speeds and resolutions. This optimization depends greatly on the type of architecture chosen for the ADC design.

2.3 Classification of ADCs

Analog-to-digital converters are often divided into three major categories based on converter speed and accuracy. Table 2.1 was adapted from [13]

Low Speed, High Accuracy	Medium Speed, Medium Accuracy	High Speed, Low Accuracy
Integrating	Successive Approximation	Flash
Oversampling	Algorithmic	Two-Step
		Pipeline
		Time-Interleaved

Table 2.1: Classification of ADCs [13]

A selection of ADC architecture types with their respective sampling rate and resolution ranges can be seen in Figure 2.4.

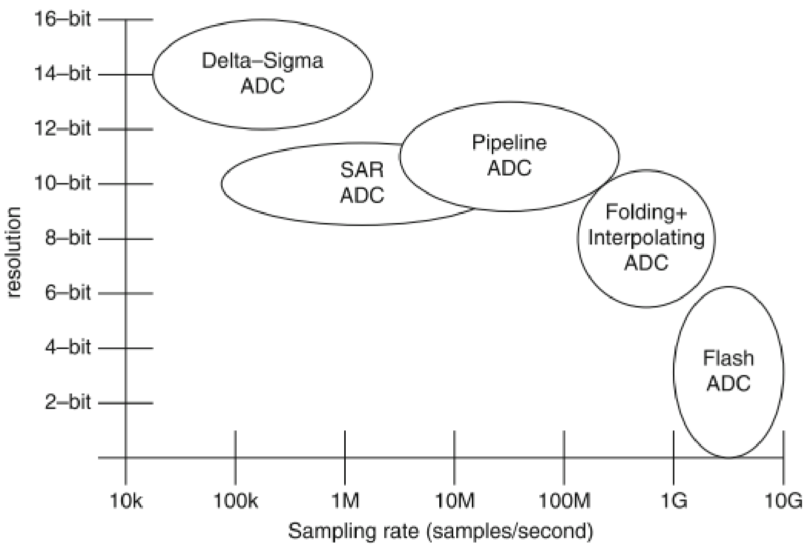


Figure 2.4: ADC Architecture Comparison [1]

2.3.1 Low Speed, High Accuracy

Some converters that are characterized by low speed and high accuracy include the integrating ADC and the sigma-delta oversampling ADC. Integrating converters are slow and their conversion times are proportional to the input voltage. Integrating ADCs require, in general, 2^Nclock cycles for N bits of resolution. A higher resolution means a slower conversion time [5].

2.3.2 Moderate Speed, Moderate Accuracy

Other converters can be categorized by moderate speed and moderate accuracy. Successive-approximation register ADCs and cyclic ADCs are both included in this classification of converters. [5]

2.3.2.1 SAR ADC

The SAR architecture algorithm is often described as being similar to a binary search algorithm. One common analogy for a binary search is looking for specific information on a page of a book. The searcher does not know the correct page and can only ask the book's owner “yes or no” questions. The search would begin by starting at the center of the book and asking if the page being searched for is a higher number than the current page. If it is, then divide the upper half of the book in half and ask the same question for the new halves until there is only one page left. The decisions algorithm for a SAR converter is shown in Figure 2.5.

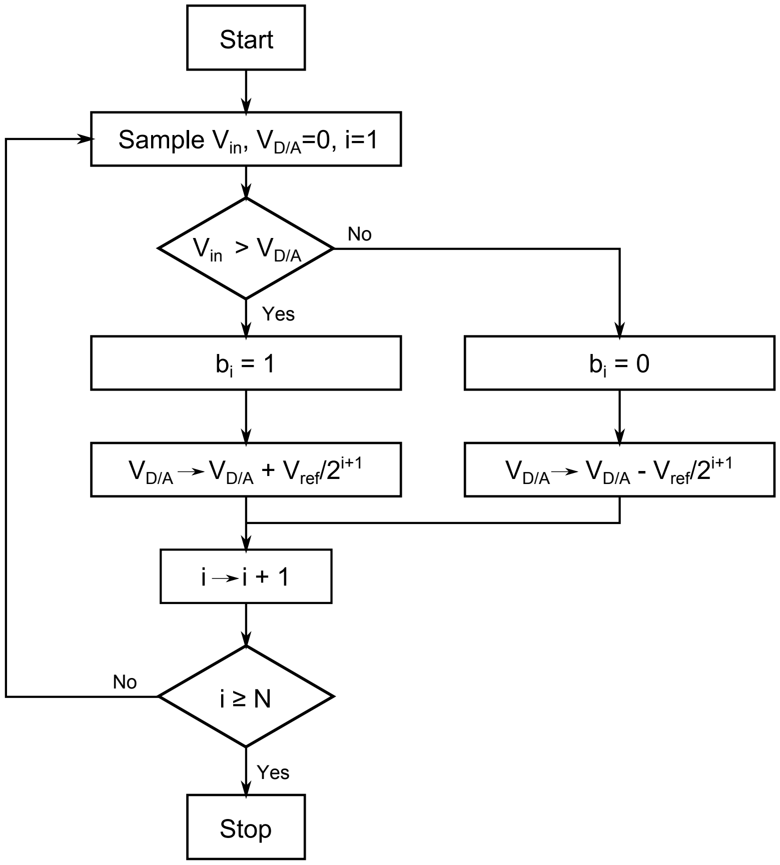

Start
Sample V_{in}, $V_{D/A}$=0, i=1
$V_{in} > V_{D/A}$
No
Yes
$b_i = 1$
$b_i = 0$
$V_{D/A} \rightarrow V_{D/A} + V_{ref}/2^{i+1}$
$V_{D/A} \rightarrow V_{D/A} - V_{ref}/2^{i+1}$
$i \rightarrow i + 1$
No
$i \geq N$
Yes
Stop

Figure 2.5: SAR Algorithm [3]

The SAR ADC follows a similar algorithm that compares input voltages and reference voltages to determine a digital output value. The main advantage that a SAR design offers is the use of only a few analog components, particularly the use of only one comparator, that results in a compact area and simpler design. The trade-off for this space is made in the maximum sampling rate. A converter with a sampling rate f_s would require the comparator, DAC and SAR logic, shown in Figure 2.6, to operate at Nf_s.

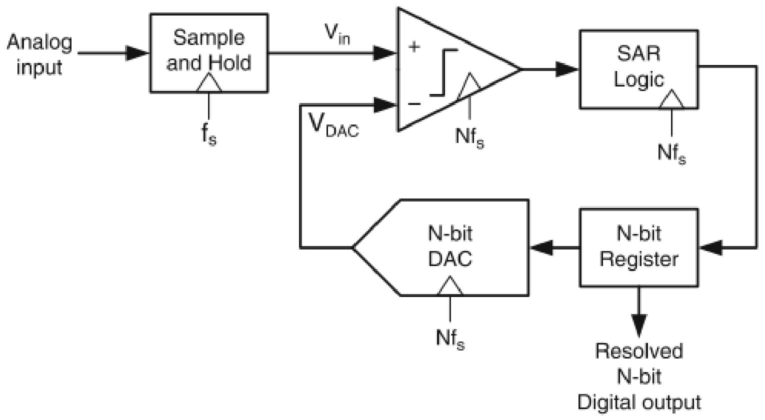


Figure 2.6: SAR ADC Topology [1]

2.3.2.2 Cyclic ADC

Cyclic ADCs also fall into this middle category. A cyclic ADC (also known as an algorithmic ADC) operates similarly to the SAR ADC. In a cyclic ADC, however, it is not the reference voltage that changes, but rather the residue is put through a gain stage and amplified. In a cyclic converter, the input is sampled and compared to a threshold voltage. A 1-bit digital output is generated and the residue generated by subtracting the output of the DAC from the original input is fed back into the sample and hold circuit. The cycle repeats for the same number of cycles as desired bits. The high level block diagram of a cyclic ADC is shown in Figure 2.7.

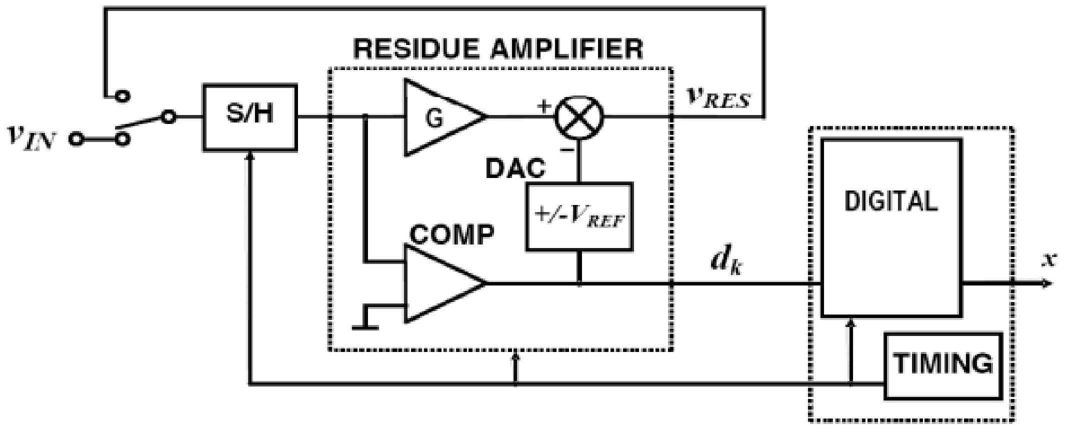


Figure 2.7: Cyclic ADC Block Diagram[17]

2.3.3 High Speed

Some converters that can be categorized as high speed and low accuracy converters are two-step, time-interleaved and flash. Pipeline ADCs can be low, moderate or high accuracy.

2.3.3.1 Flash

A flash ADC can be compared to a ruler (Figure 2.8). A ruler maps an infinite precision value length to finite precision value (e.g. 4mm). A flash ADC uses comparators to perform a similar function.

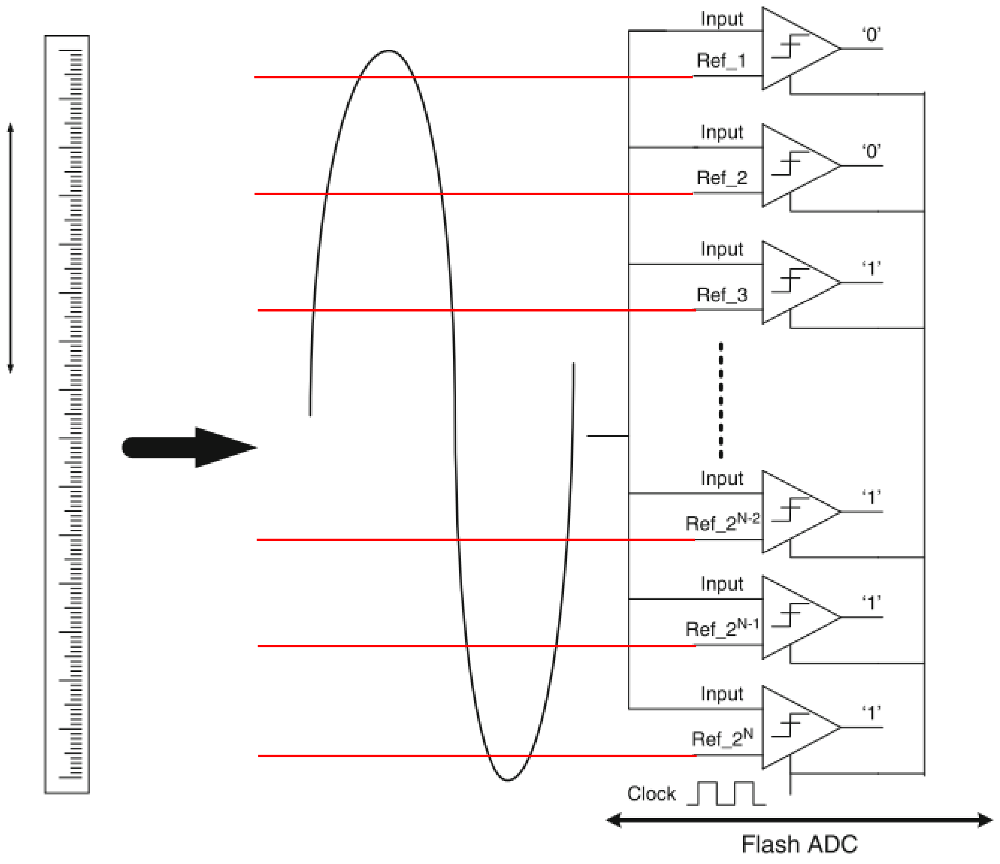


Figure 2.8: Ruler Flash Analogy [1]

A flash converter compares input (infinite precision value) to a number of fixed references to determine a binary output (finite precision value). The output of the comparators is in thermometer code. In this example, if the input is higher than the reference, the thermometer bit is one, otherwise it is zero. This thermometer code must then be translated into the equivalent binary value. The number of reference levels can be expressed as:

$$Reflvls = 2^N \tag{2.3}$$

where N is the accuracy for the ADC. A flash ADC that has 16 comparison levels will have an accuracy of 4 bits. From this relationship, it can be observed that the number of comparators required will increase exponentially compared to the increase in desired resolution. Because of this, comparators are usually used in low resolution applications. The main advantage that flash converters offer is speed. Flash comparators have the potential for conversion to take only one clock cycle. Flash ADCs are often included in the design of other ADC architectures such as a pipeline[1].

2.3.3.2 Pipeline

Pipeline ADCs are also high speed ADCs and can be capable of resolving medium to high resolutions. [1] These ADCs work by converting a signal from analog to digital in stages. Each stage converts a portion of the output resolution. The first stage converts the most significant bits (MSB) and the subsequent stages convert less significant bits until the least significant bits (LSB) are converted. The overall general architecture of a pipeline ADC is shown in Figure 2.9. Each stage has a similar structure, shown exploded in Figure 2.9. Each block contains a sample and hold block to sample the analog signal. This feeds into a small flash converter that resolves n-bits. This n-bit output is fed back through a DAC and the binary value is subtracted from the original input signal to generate a residue voltage.

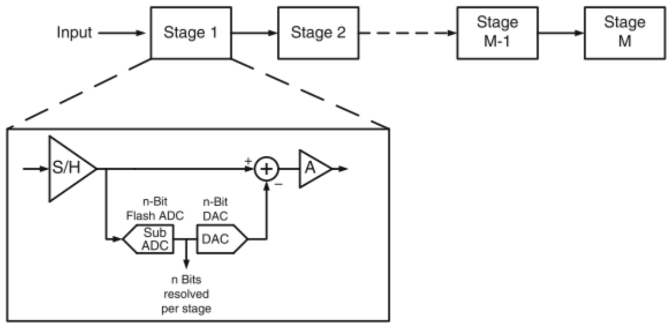


Figure 2.9: General Pipeline ADC Architecture[1]

Figure 2.10 shows a common implementation of a pipeline ADC stage. Typically, the DAC, summer, gain stage and sample and hold are implemented together in one block called a multiplying digital-to-analog converter (MDAC). The residue voltage is amplified and input into the next stage of the pipeline until the desired number of bits have been resolved.

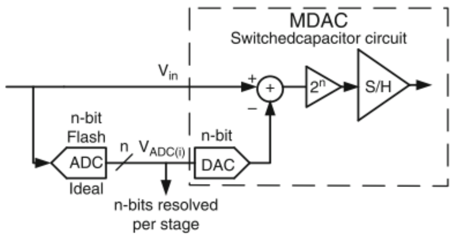


Figure 2.10: Pipeline Stage with MDAC [1]

2.4 Performance Metrics

Performance metrics are needed in order to evaluate the performance of the ADC in this work.

2.4.1 Effective Number of Bits (ENOB)

The effective number of bits of an ADC is one measure to compare different ADC designs. The ENOB is characterized by the equation

$$ENOB = \frac{SNDR - 1.76}{6.02} \tag{2.4}$$

where SNDR is the signal to noise and distortion ratio [1].

2.4.2 Figure of Merit (FOM)

A Figure of Merit (FOM) used to compare analog-to-digital converters is defined as

$$FOM = \frac{Power}{(2^{ENOB})(f_s)} \tag{2.5}$$

The FOM takes into account the power consumption of the ADC, the ENOB, and the sampling frequency f_s. A lower FOM indicates better ADC performance based on these parameters. Lower power consumption, higher ENOB and a higher sampling frequency all contribute to a lower FOM. All three of these design characteristics require design tradeoffs with one another. Increasing the sampling frequency f_s will accommodate an increased signal bandwidth, relaxes filtering requirements and can sometimes relax resolution requirements; however, increasing the sampling speed results in increased power consumption. Increasing resolution will accommodate an improved dynamic range and relax filtering requirements, but can result in increased power consumption. If trying to optimize for power consumption, compromises need to made in the design to sampling speed or resolution.

2.4.3 Differential Non-Linearity (DNL)

When the step size of an ADC's output is not equal to the ideal step size, the ADC is said to have differential nonlinearity. The DNL measurement for an ADC is classified based on amount of least significant bit (LSB) values that the actual transfer function deviates from the ideal transfer function. If the DNL is greater than 1 LSB, a non-monotonic transfer function will cause missing codes. Figure 2.11 shows the deviation of a real transfer function from the ideal.

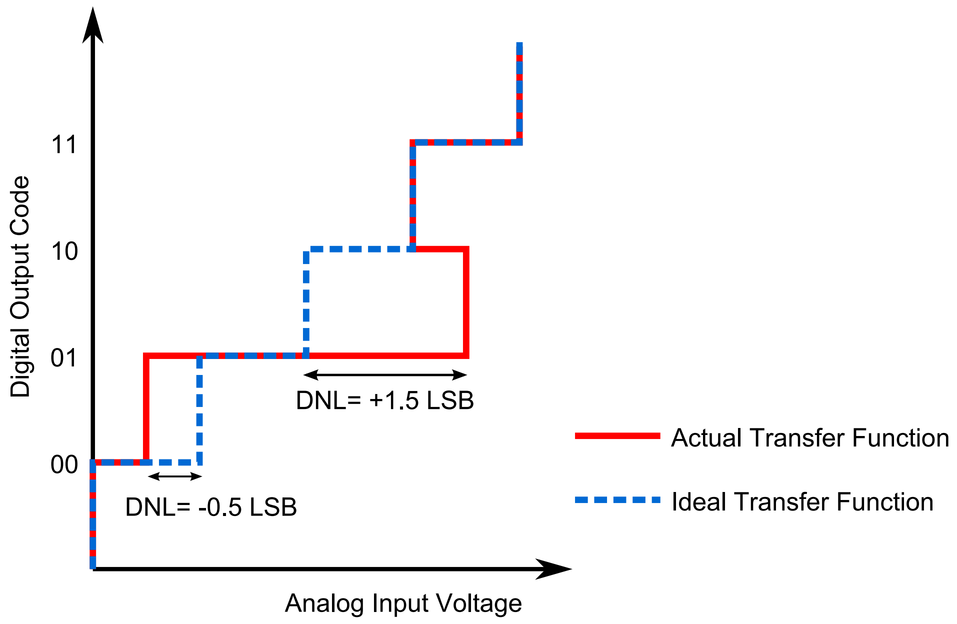


Figure 2.11: DNL

2.4.4 Integral Non-Linearity (INL)

The transfer function of an ideal ADC can be represented by a best fit line, typically either an endpoint fit or a least squares fit. An ADC that exhibits integral non-linearity will have a transfer function that is not a perfect line. The maximum difference between the actual and ideal transfer characteristic is the INL. This concept is illustrated in Figure 2.12.

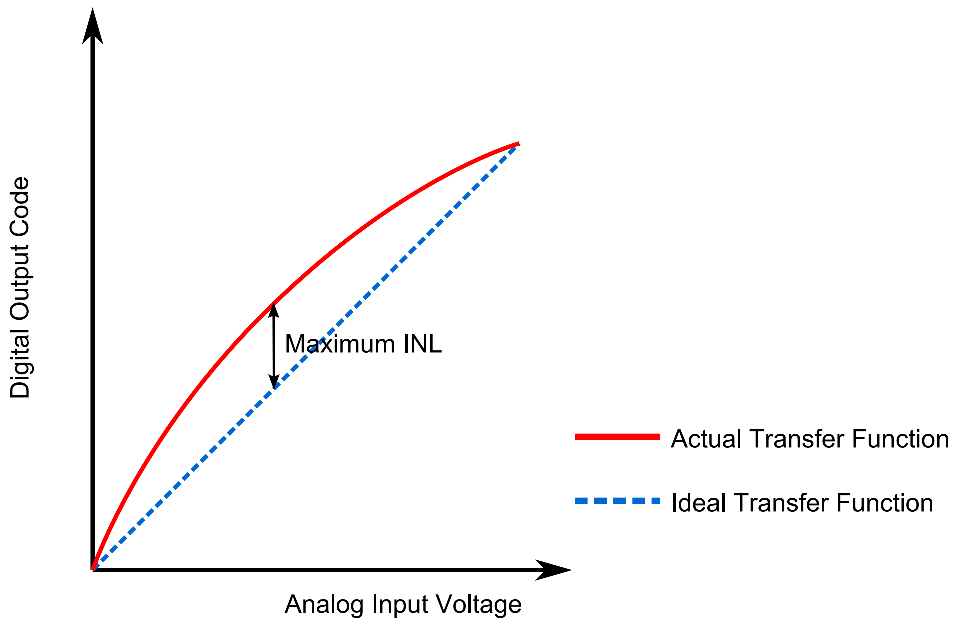


Figure 2.12: INL

2.5 Calibration

Nonlinearity in the gain stage is a common error in pipeline ADC designs. This error is caused by capacitor mismatch and low DC operation amplifier gain, but the exact error is typically not known by the designer. In order to compensate for this nonlinearity, various calibration techniques are used.

2.5.1 Foreground Calibration

In a foreground calibration scheme, the unknown errors are estimated by interrupting the operation of the ADC and then injecting a known signal. The expected output is compared to the actual output to measure the error [16]. Once the error is acquired, Least Mean Square (LMS) algorithms can be used to correct for the error.

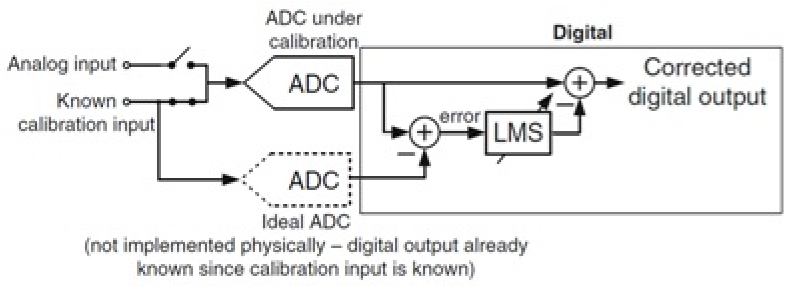


Figure 2.13: Foreground Calibration Block Diagram

As shown in Figure 2.13, analog input signal is fed into the actual ADC and a known signal is fed into the ideal ADC. Since it is impossible to implement an ideal ADC, this component is simulated digitally. Another digital component is used to calculate the error between the actual output and the ideal output. This same digital component will then correct the digital output for this calculated error. The main advantage of using foreground calibration is that one can achieve the corrected digital output in a few clock cycles. However, the operation of the ADC is interrupted during calibration. This interruption is impractical in some applications.

2.5.2 Background Calibration

Background calibration technology can correct errors of ADC circuits without interrupting the operation of the ADC. Methods of background calibration can be analog or digital and have a variety of implementations.

2.5.2.1 Bootstrapped Digital Calibration

The bootstrapped digital calibration scheme is one of the famous calibration methods as it can reduce the calibration convergence time [23],[3]. In this case, the ADC is utilized to calibrate the DAC and vice versa. Bootstrapped digital calibration includes analog circuits in the part of the calibration process to more accurately track the voltage and current samples. The addition of these analog circuits increases the overall power consumption of the ADC.

An accurate, constant gain and signal dependent gain are required for bootstrapped calibration [3]. These two gains are known, however, so an initial estimate of the gain values is required. The estimation of the constant gain is then updated 1024 times depending on the measured positive and negative thresholds of the residue characteristic curve. After updating the constant gain estimate, we need to update the signal dependent gain. To update the signal dependent gain, the linear and nonlinear ADC transfer characteristics are used. The signal dependent gain is updated 256 times. The number of update times, 1024 and 256 are selected analytically, but they are only the minimum number of times required for convergence [3]. As the two gain values are repeatedly fed into the ADC and DAC of each stage, the gains are constantly being updated, and eventually these two values will converge, resulting in successful calibration.

2.5.2.2 1.5-bit Stage ADC Architecture

A 1.5-bit stage architecture uses two approximately symmetrical analog voltage levels to produce an implementation with increased bandwidth and redundancy between stages. The 1.5-bit stage pipeline ADC architecture achieves greater bandwidth by using a lower interstage gain [21]. Due to this low gain requirement, we can realize low cost production and higher speed. Each stage generates an output of two bits in which the bits can only have the value of 00, 01, and 10. The output is determined by comparators at two symmetrical decision levels that make up a sub-ADC block of a pipeline architecture. Because of the following gain of 2, these two levels must be within the range of $\pm\frac{V_{REF}}{2}$, where $\pm V_{REF}$ are the maximum and minimum values of the signal. The choice of these reference value is not highly critical in the design, but because they must lie within the range of $\pm\frac{V_{REF}}{2}$ the decision values are often chosen to be $\pm\frac{V_{REF}}{4}$. These decision levels are designated as +1, 0, and -1 and are used in an implementation called a Redundant Signed Digit (RDS). The redundancy comes from the 0.5 bit overlap between stages. When the stages are summed, the carryover from the previous stage creates a redundancy and error correction.

2.5.2.3 Murmann's residue gain error correction

Murmann's residue gain error correction calibration method starts with adding a logic block to the output of the sub ADC block in each stage of the pipeline ADC [11]. This logic block provides two different residue characteristics that generate Figure 2.14.

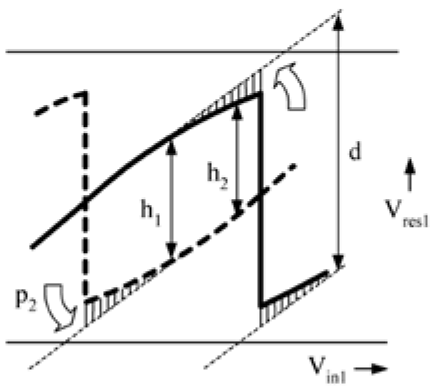


Figure 2.14: Difference Between Two Residue Characteristics

The distance between one residue plot versus the other can show the linearity of the ADC. In this case, h_1 represents the ideal distance while h_2 represents the nonlinear distance. The goal is to apply an adaptive routine to correct the error between h_1 and h_2 such that error will converge to zero.

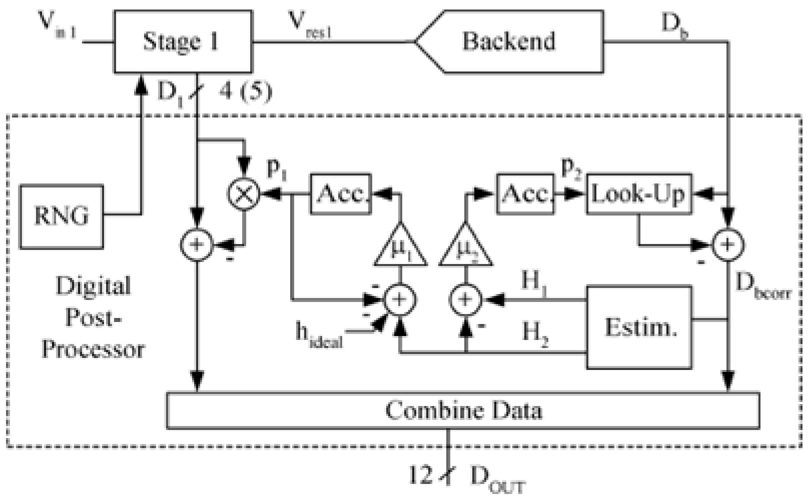


Figure 2.15: Error correction algorithms by using PDF

First, the probability density function of the residue characteristic is calculated to estimate h_1 and h_2 by using a random number generator. With the estimations of h_1 and h_2, we can calculate the error. Then, the LMS algorithm is applied to force the error to zero. Once this is achieved that one can adjust the parameters p_1 and p_2 to force the output of Stage 1 and the backend stage to be linear. The main advantage of this technique is that it can achieve low power consumption.

2.5.2.4 Split ADC Architecture

The split ADC architecture is known for being able to calibrate residue gain error over a short period of time [2]. It can also digitally correct DAC errors in pipeline ADCs. In the split ADC architecture, there are two ADCs with the same resolution. The only difference between them is the residue transfer characteristic. Those two ADCs are placed in parallel and are applied with the same input signal. The following diagram shows how the split ADC architecture is used in a pipeline ADC.

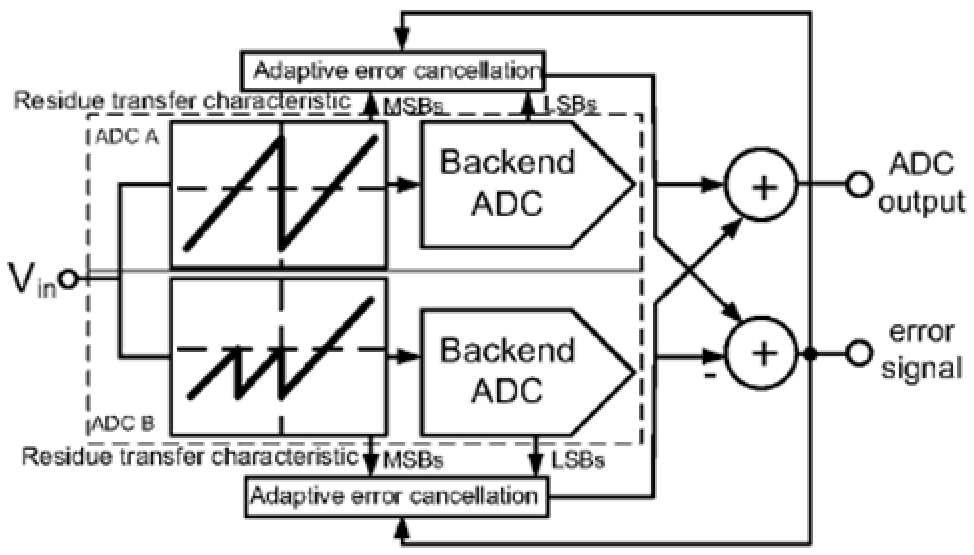

Adaptive error cancellation
Residue transfer characteristic
MSBs
LSBs
ADC A
Backend ADC
ADC output
V_in
Backend ADC
error signal
-
ADC B
Residue transfer characteristic
MSBs
LSBs
Adaptive error cancellation

Figure 2.16: Split-ADC utilized in a pipeline ADC

As shown in the Figure 2.16, the same V_{in} is input into the two split ADCs, ADC A and ADC B. However, the outputs of the two ADCs are different for every input due to their different residue transfer functions. The difference between the outputs of those two ADCs is the error of the residue amplifier. Using this difference, the adaptive error cancellation can be processed to correct the residue amplifier gain error. Then the outputs from each of the adaptive error cancellation block are added to get the final output of the pipeline ADC.

In [2], to implement a 12-bit pipeline ADC, the authors incorporated two stages in each of the split ADCs in their design. The first stage consists of a 4-bit pipeline stage and the second stage consists of a single 10-bit flash ADC. In this work, only the first stage is calibrated and the second stage does not need to be calibrated. Even though the goal is to implement a 12-bit ADC, they included two extra bits to achieve more accuracy in error correction. A different residue transfer characteristic in the two ADCs in the split ADC architecture can be acquired by offsetting one residue transfer characteristic curve with respect to the other [2]. Due to the residue amplifier gain errors, the slopes of backend codes will not be similar. Therefore, the difference between the outputs of two ADCs is not equal to zero. By using that difference we can adapt a corrective term to fix the residue amplifier errors, which would also calibrate the DAC's nonlinearity.

Another split pipeline ADC architecture is described in [12]. In this paper, similar to the previous design, the ADC is split into two identical ADCs, processing the same input but producing different outputs as shown in Figure 2.17.

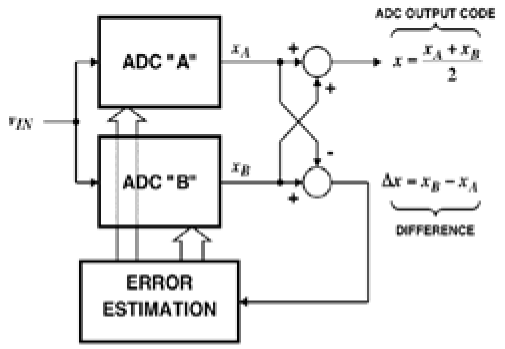


Figure 2.17: Split ADC Architecture

The average of the two outputs becomes the output of the ADC. The difference between the two outputs is used to calibrate the ADC. If the difference between two outputs is zero, there is no error and the ADC is calibrated perfectly. If the difference is nonzero, that difference is used to adapt the error corrective term and update the calibration parameters in each ADC to achieve an error of zero. Finally, the advantage of the split architecture is its fast calibration convergence. [12]

Chapter 3

System Level Design

To perform a system level design we will define our specifications, provide a system block diagram and take a closer look at the circuits that need to be designed in each stage.

3.1 Design Specifications

The analog to digital converter was designed to be part of a self-calibrating split non-linear pipeline ADC. The basic specifications for the design are outlined in Table 3.1.

Specifications	
Circuit Type	Integrated Circuit
Maximum Size	1mm^2
Process Type	0.18um
Resolution	12 bits
Throughput	20 MS/s
Power	1mW
Other Specs	Fully Differential: 1V_{pp}

Table 3.1: Pipeline ADC Specifications

This pipeline ADC was designed for the 0.18um Jazz Semiconductor process. This process allows for a 1.8V supply voltage. Because of the small amount of headroom allowed by this supply voltage, the signal swing the ADC can handle is 1V_{pp}. This differential signal reduces second order distortion in the sample and hold circuit and doubles the input range from that possible in a single ended system.

The ADC will resolve up to 12-bit accuracy with a throughput of 20 MS/s. Each stage resolves 1.5 bits, creating redundancy between stages. Ten 1.5 bit stages, ending with a 2-bit flash ADC, are used to generate a final 12-bit digital output. Each 1.5 bit stage is designed to have an identical residue characteristic for successful calibration. The throughput of the system is only dependent on the time of a single stage. There is latency across the ADC, but the throughput remains high.

Several aspects of the design reduce power consumption. Large flash ADCs consume high amounts of power since the number of comparators required roughly doubles for each additional bit of resolution. The small resolution of each stage allows the number of required comparators to scale linearly for additional bits of resolution. The stages are also scaled down in size for the less significant bits. This reduces the power consumption of the smaller stages. This pipeline ADC also uses an open loop gain stage. Using an open loop gain stage rather than a closed loop system can reduce power by up to 50-70%. [16]

3.2 System Block Diagram

The overall system block diagram of the ADC is shown below in Figure 3.1.

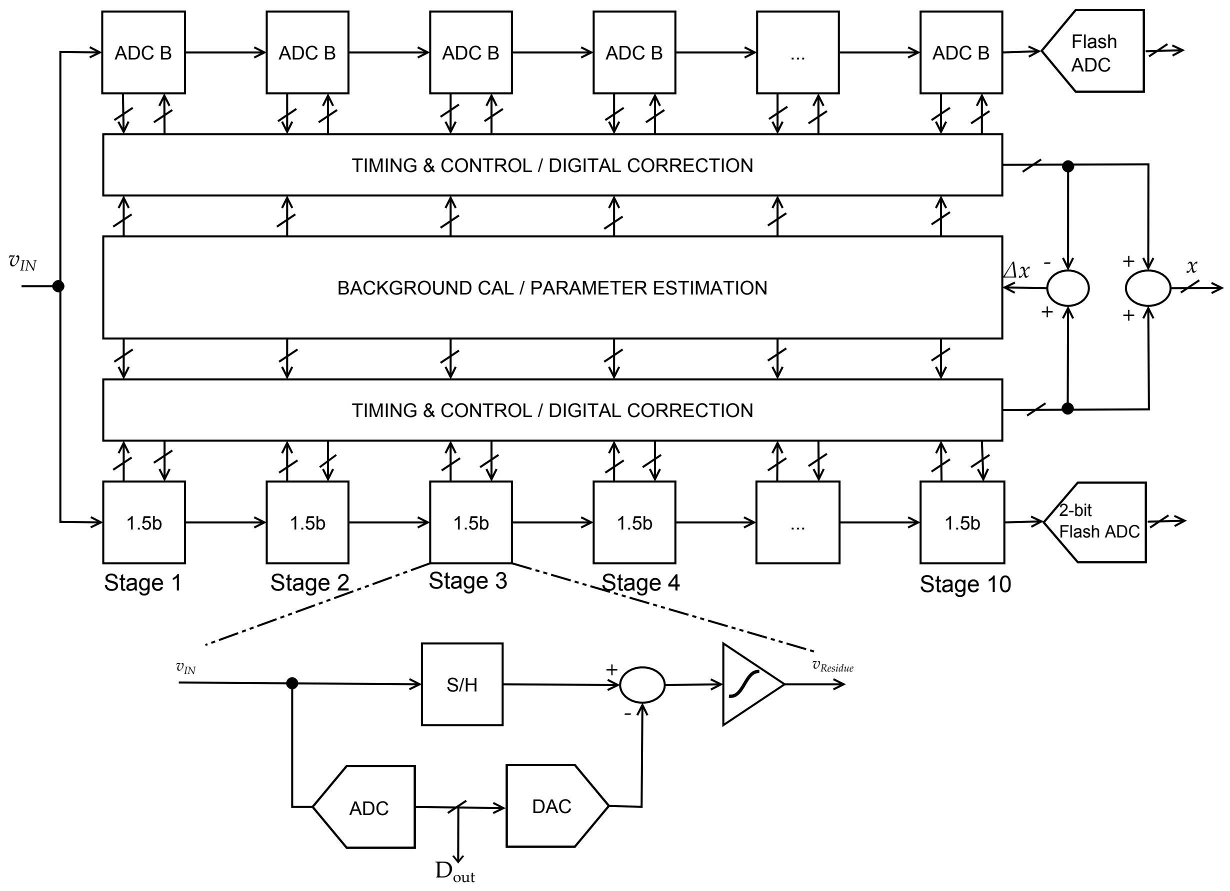


Figure 3.1: System Block Diagram

This ADC was implemented for use in a split architecture that includes two separate pipeline ADCs. Two different residue plots are required for each side of the ADCs for the calibration to be successful. Each stage is connected to the digital calibration block. The calibration algorithm feeds digital correction information back into the system based on the difference between the outputs of the two ADCs. The scope of this project included the design of ADC A. The 0.5-bit overlap between consecutive 1.5-bit stages creates redundancy which is used in calibration to account for ADC linearity and interstage offset[20].

3.2.1 Stage Block Diagram

The block diagram of each stage of the ADC is shown below in Figure 3.2.

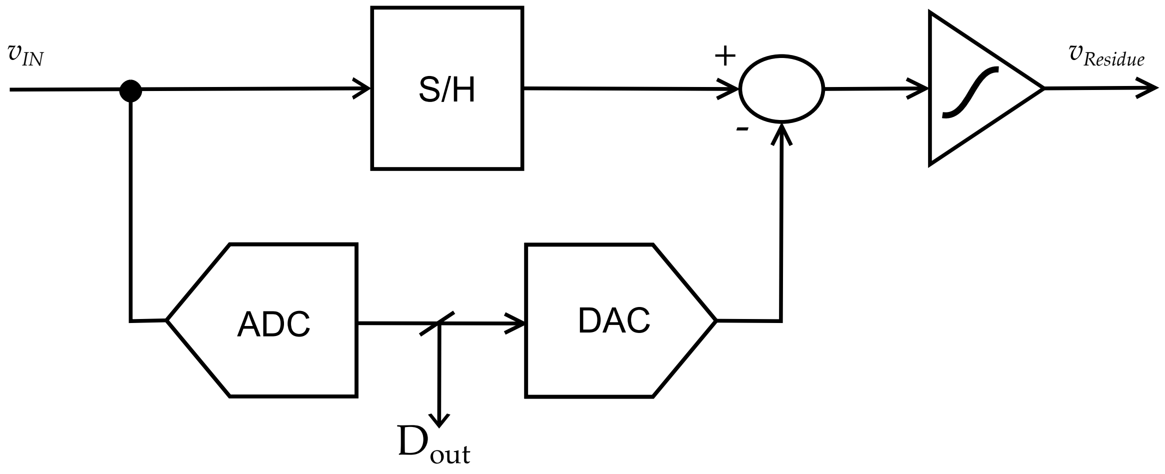


Figure 3.2: Stage Block Diagram

Each stage is made up of five main parts as shown above; the sample and hold, 1.5 bit flash ADC, DAC, summer and a residue amplifier. During Phase 1, the signal is sampled by the sample and hold circuit. Just before Phase 1 is completed, the comparator holds the value of the input and generates a decision, which is sent to the switch capacitor circuit's DAC after Phase 1 is finished. The input value is then sampled across the capacitor and the DAC value is subtracted based on the comparator decision. This new value is then amplified by 2^N-bits, in this case 2 since 1 complete bit is resolved in each stage. The output of this residue amplifier feeds into the subsequent stages and each stage evaluates as described.

It was seen in [16]that the S/H, summer and DAC can all be combined in a switched capacitor network. This simplifies the stage block diagram to only three main blocks: switched capacitor, flash ADC and residue amplifier. Not shown in the block diagram above are a few key blocks that are fundamental to circuit operation. These blocks are the bias circuitry for the residue amplifier, the output drivers for each of the digital decisions, and a timing block to control the timing of the switches. If we add these blocks, along with simplifying the stage block diagram we obtain the following, the result can be seen in Figure 3.3.

- Switch Capacitor Array
- Open Loop Residue Amplifier
- Bias Circuitry
- Comparator Network
- Timing Circuit
- Output Drivers

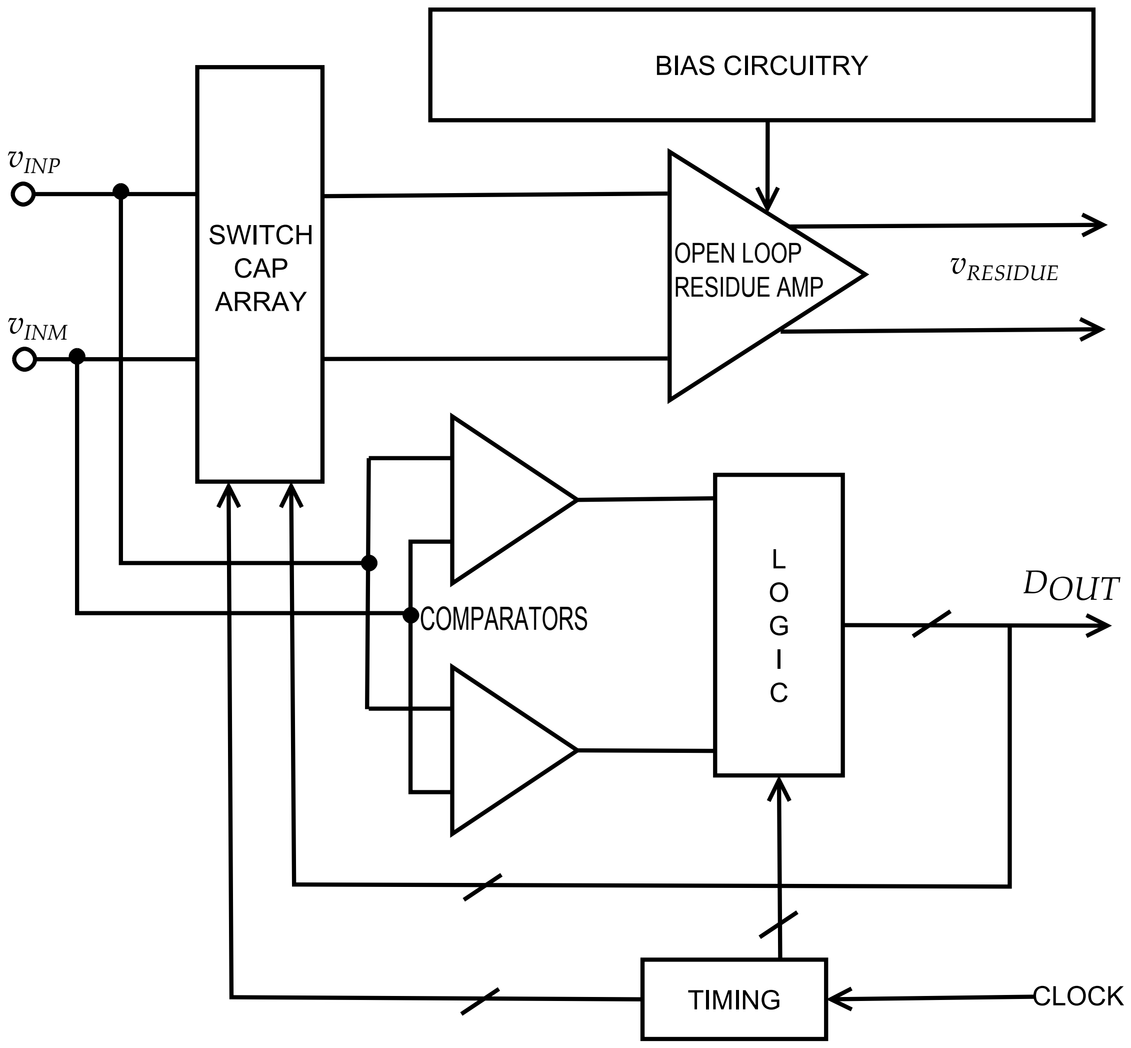


Figure 3.3: Project Block Diagram

3.2.2 Timing

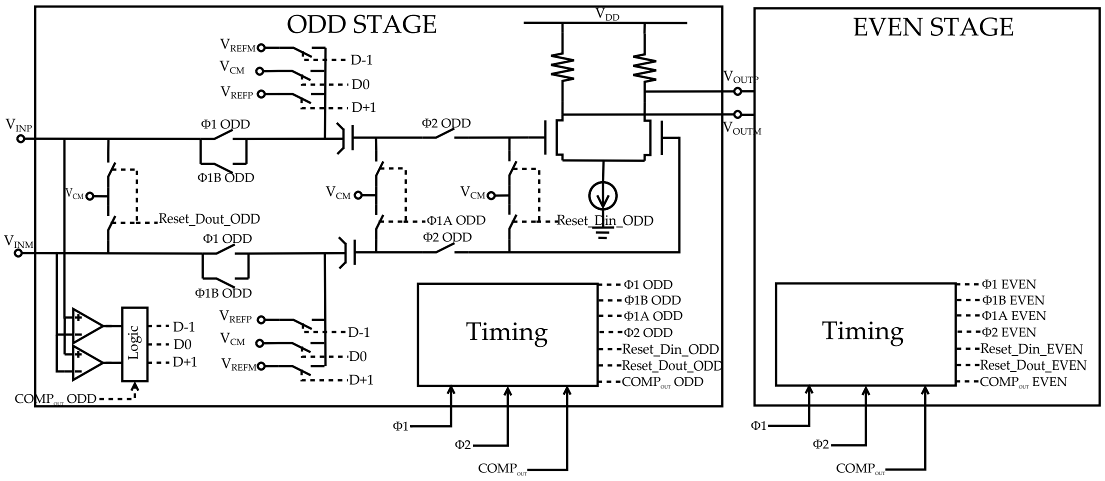


Figure 3.4: Detailed Switch System

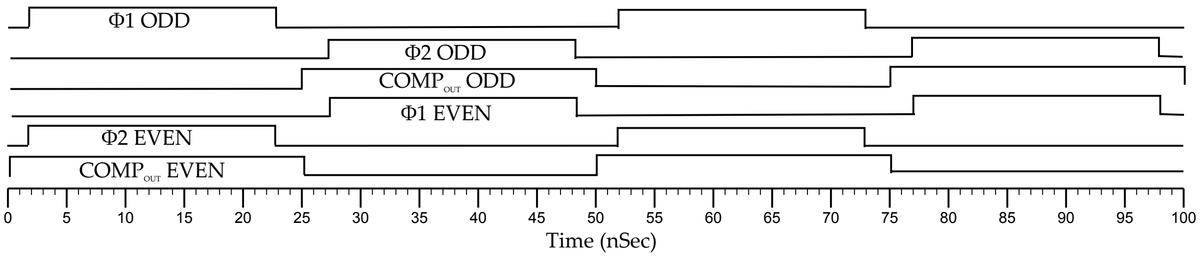


Figure 3.5: System Timing

Timing is critical when designing mixed-signal highly switch dependent circuits such as ADCs. Specifically, when controlling the switch capacitor array which connects to all of the other blocks and must control the sequence in which information is sent and received by the switched capacitor circuit. A switch diagram of the entire system is shown in Figure 3.4.Three main operations must occur. First the circuit needs to sample the input, Phase 1, then hold its value, during which the comparator makes its decision and then needs to switch the correct decision in the switch capacitor circuit, and finally the circuit must send its value to the residue amplifier, Phase 2. The time that the residue amplifier has to settle should be the same amount of time that the input is sampled, which means Phase 1 and Phase 2 must be equal lengths. The reason for this is that while the value is being amplified by the first stage, the next stage must be sampling at the same time. This means that for the even stages there will be one timing circuit and for the odd another, one with Phase 1 first and the other with Phase 2 first.

Since the speed of our circuit is 20 MHz, this leaves 50ns for each stage. This 50ns should be split between the two phases. Equally splitting the time will not work since there much be time between the two phases for the comparator to make its decision and send its value to the switch capacitor. Figure 3.5 is a simplified timing diagram for the overall circuit, displaying both the even and old timings.

3.2.3 Stage Scaling

The stages of the design are scaled to reduce power consumption and meet noise requirements. The first stages need to be the most noise resistant since they resolve the most significant bits. [17] The scaling architecture is shown in Figure 3.6; each stage is halved in size until it reaches the minimum capacitor size of 125pF. This allows for each of the stages to be made up of the smallest stage and the largest stage that is made up of 8 of the smallest stage has the capacitance value of 1pF. This modular design is done to reduce layout time by being able to place stages in parallel to create each of the differently sized stages.

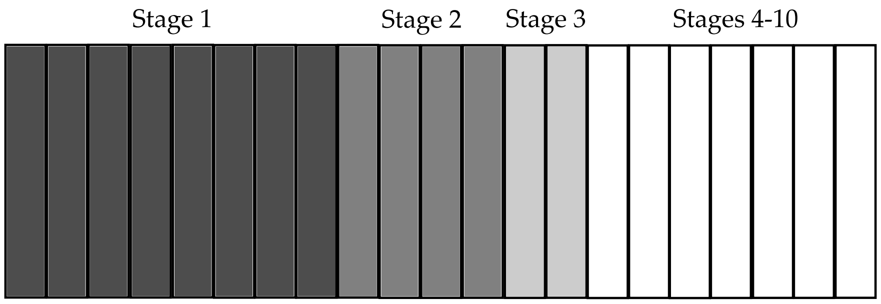


Figure 3.6: Scaling

Chapter 4

Switched Capacitor Array

4.1 Introduction

The switched capacitor array performs the functionality of the sample and hold, digital-to-analog converter, and summation blocks, highlighted in gray in Figure 4.1. The combined blocks perform the operation $V_{IN} - V_{DAC}$. The switch capacitor array also dictates the timing of the overall ADC design. The switched capacitor array input is the output of the residue amplifier of the previous stage.

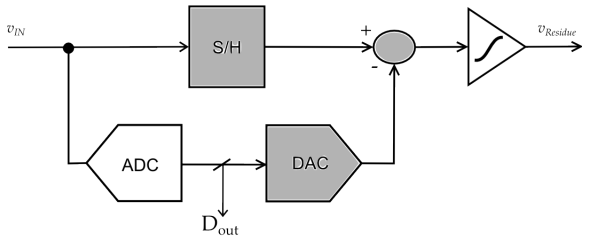


Figure 4.1: System Block Diagram

The switched capacitor array is comprised of capacitors and MOSFET switches. The circuit uses charge conservation and Kirchhoff's current laws to perform the subtraction needed in the ADC. A simplified, single-ended example of a switched capacitor network will illustrate the functionality of the circuit. The implementation used in the pipeline ADC was fully differential.

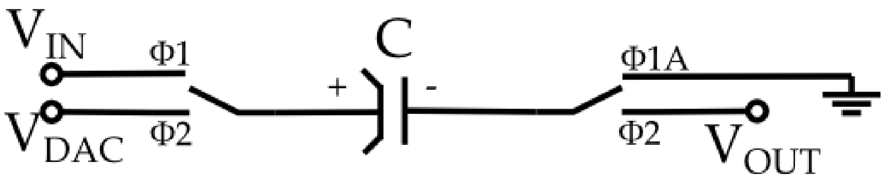


Figure 4.2: Switch Capacitor Theory

The switch capacitor circuits operates in two phases, phase one Φ1 and phase 2 Φ2. The operation of two switches is dictated by these phases. The switch connected to the top plate of the capacitor is switched to V_{IN} during Φ1 and the DAC voltage during Φ2. The bottom plate of the capacitor is connected to ground in phase one and connects to the output during phase two. This is shown below in Figure 4.2 and also notice Φ1A on the top plate, A meaning advanced, refer to Figure 4.6. This signal is advanced, that is it triggers in advance of the phase one signal, in order to reduce charge injection.

Phase one is shown below in Figure 4.3. During this phase the input voltage is charged across the capacitor while the other end is held at ground. In the differential implementation, it is switched to the common mode voltage instead of ground.

$$Q = CV = CV_{IN}$$

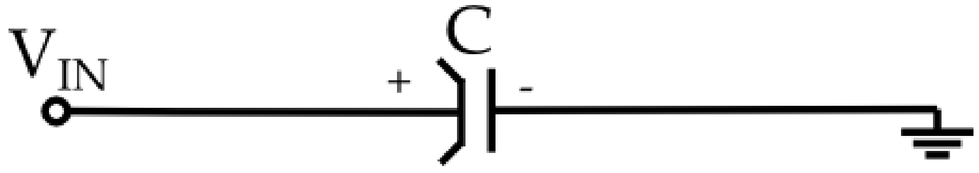


Figure 4.3: Switch Capacitor Theory: Phase One

Phase two is shown in Figure 4.4. During this phase the top plate is switched to the output of the switched capacitor array which is the high impedance node of the input of the differential pair. The bottom plate is then switched to the DAC value. When KVL is used to determine the value of V_{OUT}, the value is equal to the DAC voltage V_{DAC} subtracted by the value across the capacitor V_C. The voltage across the capacitor is V_{IN}.

$$V_{OUT} = V_{DAC} - V_{IN}$$

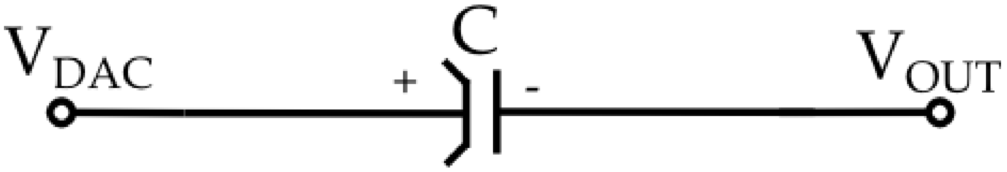


Figure 4.4: Switch Capacitor Theory: Phase Two

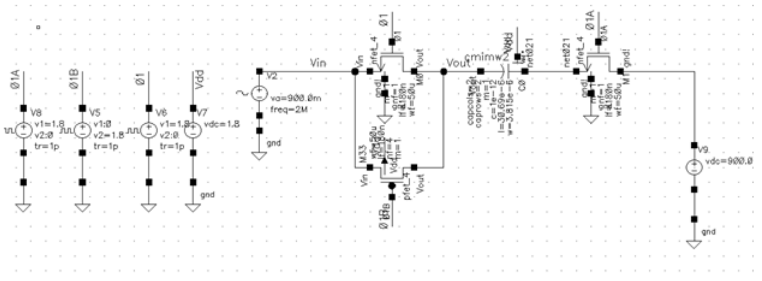
This output behavior is not exactly that required. In order to get the required $V_{IN} - V_{DAC}$ operation, when implementing the circuit differentially, the outputs are inverted by switching V_{OUTP} and V_{OUTM}.

4.2 Design and Simulation

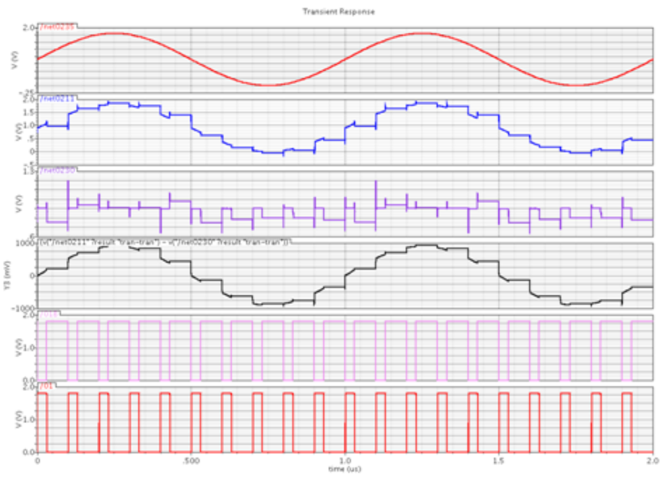
4.2.1 Sample and Hold

The sample and hold circuit was designed first. This circuit follows the input and at determined time intervals holds the voltage across a capacitor. The sample and hold (S/H) circuit captures a sampled value of the continuous input signals and allows for the rest of the circuit to use the held value stored on the capacitor.

A basic sample and hold circuit can be realized with a simple single NMOS or PMOS MOSFET acting as a switch and a capacitor connected to ground. A single NMOS or PMOS switch will generate imperfect logic levels because of the threshold voltage of the device. A complementary switch is able to produce more ideal characteristics. This complementary circuit is called a transmission gate. In this design, the PMOS switch in the transmission gate is four times larger than the NMOS switch to account for the lower mobility in the PMOS device.



(a) Schematic



(b) Simulation

Figure 4.5: Top Plate Switching

The capacitor size determines the noise of the circuit. The capacitor was chosen to be 1pF. The RMS noise is $\sigma_v = \sqrt{\frac{KT}{C}} = 64uV$. The noise on the differential design is $64uV \times \sqrt{2} = 90uV$. The noise of the differential design combined with the other noise sources creates a total of about $200uV$, yielding a signal to noise ratio of $SNR = \frac{\mu}{\sigma} = 20log(\frac{.3535V}{200uV}) = 64.9dB$. The ideal SNR of a 12-bit ADC is $SNR_{ideal} = 6N + 1.78 = 73.78dB$. To improve the SNR, the capacitor value has been doubled to 2pF resulting in a signal to noise ratio of 67dB, which is closer to the ideal value of 73.78dB. The increase in capacitor size will improve the signal to noise ratio but at the expense of die area.

To reduce charge injection, a "top-plate switching" technique was implemented. A switch on the top plate of the capacitor is added. The top plate of the capacitor is switched to common mode slightly before the signal controlling the transmission gates, Φ1. This advanced signal is Φ1A. This timing is shown in Figure 4.6 and Figure 4.5 shows the circuit implementation and simulation results. The bottom plate of the capacitor shown in blue in Figure 4.5b shows what the circuit would look like without the top plate switching. The black waveform, taken across the capacitor shows the reduction of charge injection.

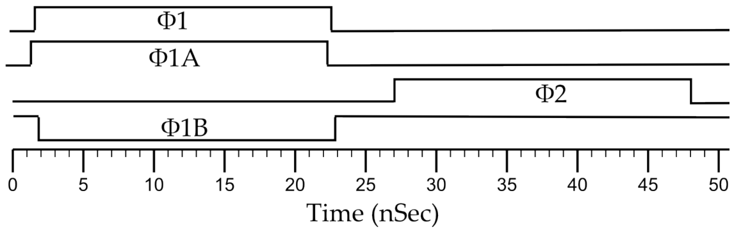
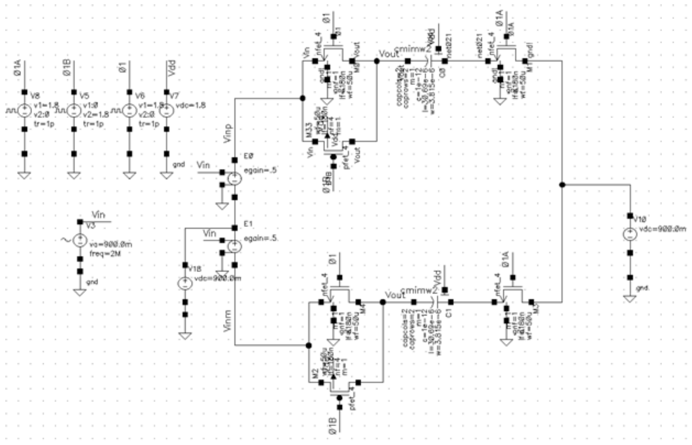
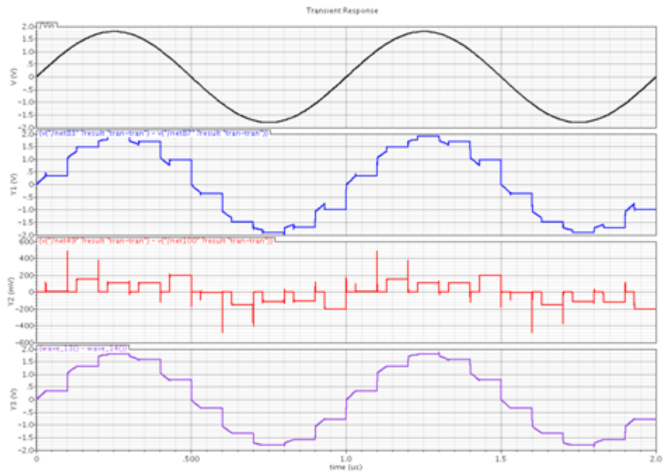


Figure 4.6: Switch Capacitor Basic Timing

In order to reduce noise, the switched capacitor array was modified to be fully differential. Figure 4.7 shows the schematic and simulation of the switched capacitor circuit implemented differentially. The schematic is the complete sample and hold circuit. The simulation yields a reduction in charge injection, as seen in Figure 4.7b.



(a) Schematic



(b) Simulation

Figure 4.7: Differential Sample and Hold

4.2.2 Digital-to-Analog Converter

The DAC's main function is to output an analog voltage based on the comparator decisions. The ideal residue plot in Figure 4.8 is the transfer function that must be created by the switched capacitor circuit. Where the decisions are made is determined by the comparator, but the shape is formed based off the DAC voltages.

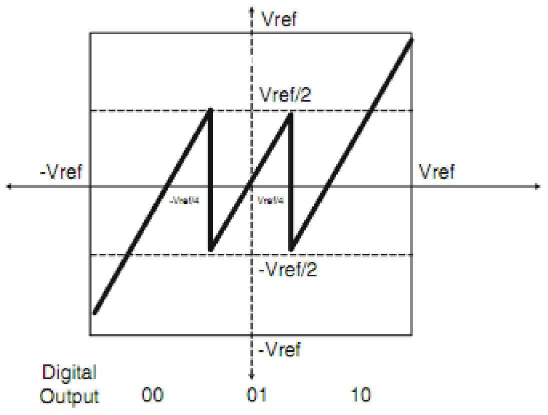


Figure 4.8: Ideal Residue Plot

From the residue plot and the basic concept of the DAC, the reference values used are $\pm\frac{V_{REF}}{4}$. Figure 4.9 helps show how the reference values are implemented. For the zero decision both DACs are switched to the common mode voltage. For the plus one decision (D+1), the DAC of the positive path is switched to V_{REFP} and the negative path is switched to V_{REFM}. The opposite is done for the minus one decisions (D-1).

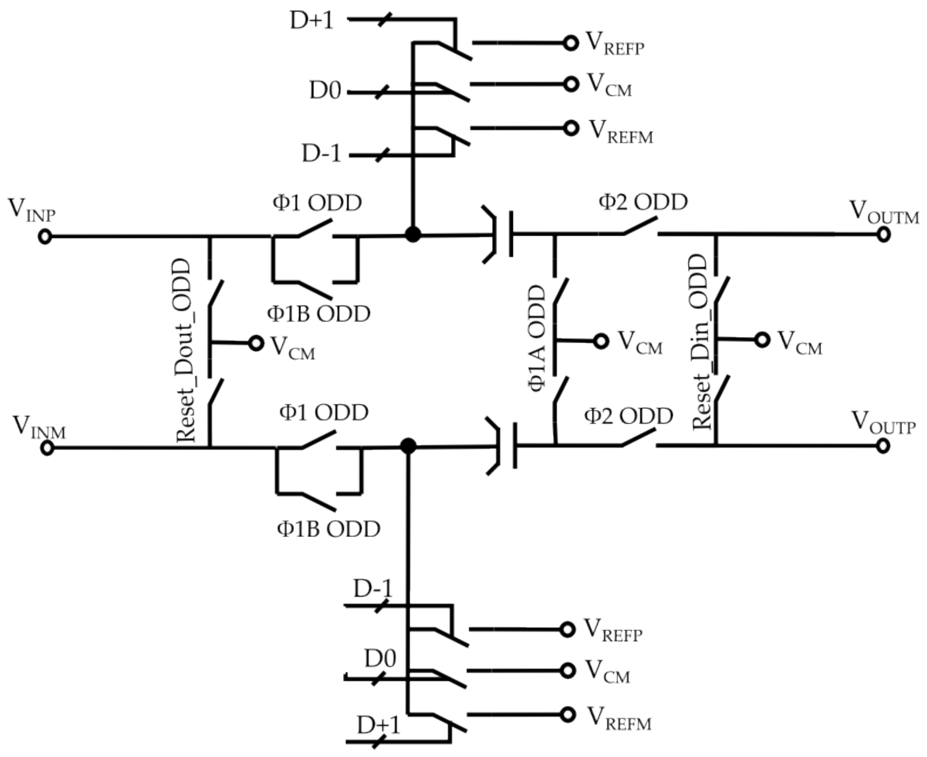
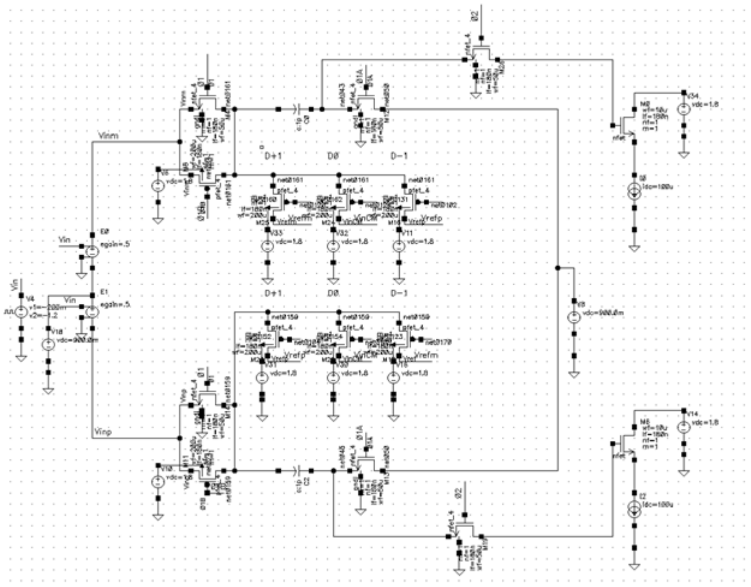


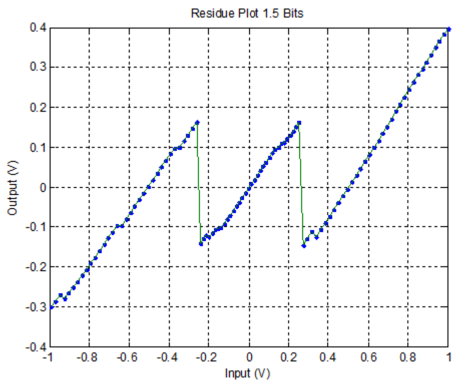
Figure 4.9: Switch Capacitor Array Switch Diagram

It was found that $V_{CM}$was needed to be 1.5V. The differential pair Design section, 5.2, details the reason for this common mode. Once this value was determined, this limited our input swing to only $500mV_{pk}$ making $V_{REF} = 500mV$, $V_{REFP} = V_{CM} + \frac{1}{4}V_{REF} = 1.625V$ and$V_{REFP} = V_{CM} - \frac{1}{4}V_{REF} = 1.375V$. This high common mode voltage required the use of PMOS for the DAC switches in order to keep the switches in the correct operation region.

In the final design, the comparator decisions control the DAC switches. Initial simulations were run with sources simulating the decisions from the DAC to isolate and test only the switched capacitor functionality. The residue plot show in Figure 4.10b exhibits non-linearities in the switch capacitor stage even without the non-linear gain stage. Note, the residue plot does not include the gain stage.



(a) Schematic



(b) Simulation Residue Plot

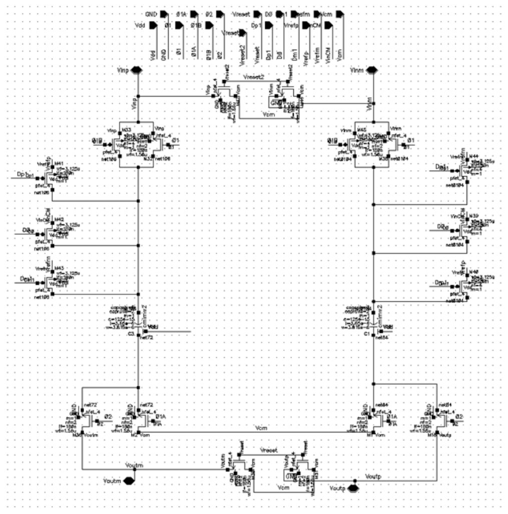
Figure 4.10: Switch Capacitor w/ DAC

4.2.3 Reset Switching

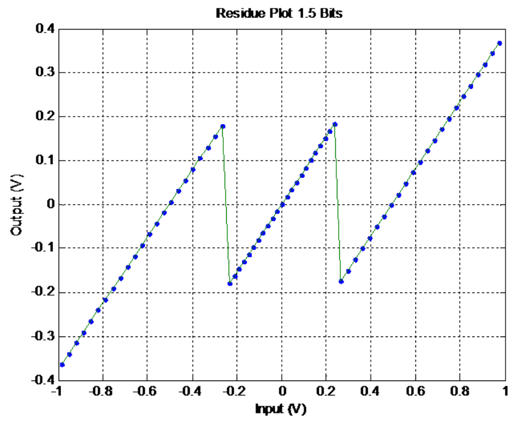
Reset switching was used to reduce non-linearities caused by charge stored in the parasitic capacitors in the differential pair. The reset switches were placed at the inputs and the outputs of the switched capacitor and switch the differential signals to common mode. This allows the circuit to always start from the same value and eliminate the memory of the last value on the residue amplifier.

The reset switch at the input of the switch capacitor controls the output of the previous stage, RE-

SET_DOUT. The timing for this switch is on during Φ2 while the circuit is evaluating. At the output, there is RESET_DIN that resets the input of the residue amplifier. As a result, shown in Figure 4.11b, the residue plot of the circuit is now linear before the gain stage is applied. Detailed information regarding the timing of these switches is located in the timing Design section, 9.2.



(a) Schematic



(b) Residue Plot

Figure 4.11: Full Switch Capacitor w/ Reset Switching

After the reset switches were added, the circuit was combined into a block for full circuit simulations. This symbol is shown below in Figure 4.12.

4.3 Layout

The layout of the switched capacitor network is affected by the scaling techniques used in the design. In order to facilitate reusability of design, the larger stages that resolve the MSBs are composed of the smallest stages connected in parallel. This smallest stage is scaled to $\frac{1}{8}$ of the largest stage.

The layout was designed to facilitate connection of stages in parallel horizontally and maintain the signal path from top to bottom. The fully differential nature of the circuit led to the use of symmetry with respect to the vertical center axis so that the positive path is on one side and the negative is on the other. The final layout is shown in 4.13. Extracted simulations were run for this layout and functionality of the extracted design was verified.

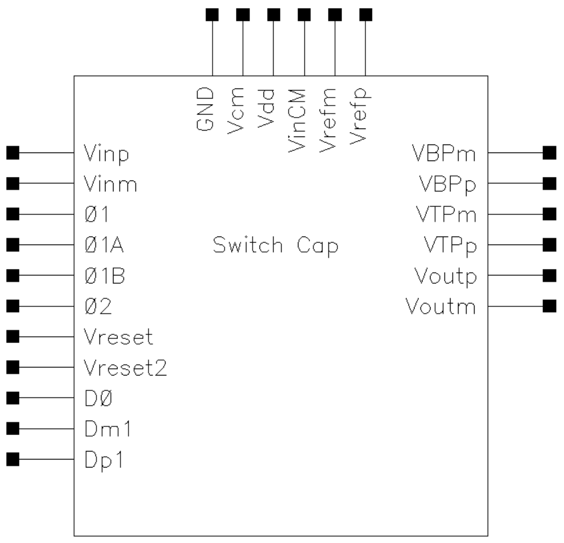


Figure 4.12: Switch Capacitor Circuit Block

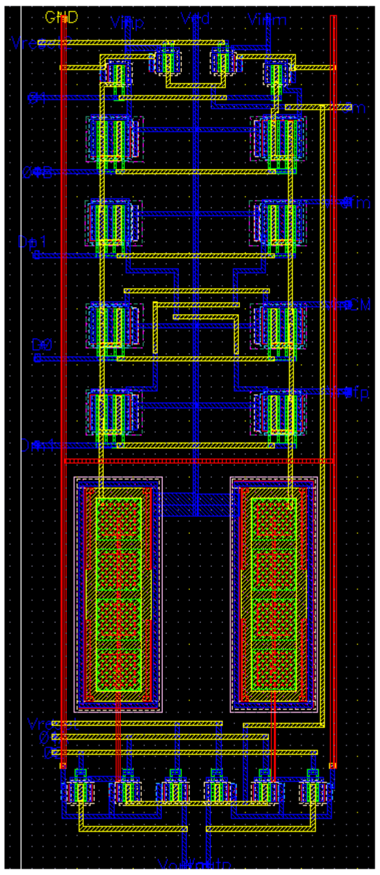


Figure 4.13: Switch Capacitor Array Layout

Chapter 5

Residue Amplifier

5.1 Introduction

The purpose of the residue amplifier is to amplify the residue at each stage in order to scale it back to the range of the sub ADC block. With the amplified residue value, the comparator in the sub ADC of the next stage can make the right decision and acquire the correct digital value at the output of the sub ADC. The residue amplifier is highlighted in the following block diagram, Figure 5.1. The input of the residue amplifier is the residue produced by the switched capacitor based on the comparator decision. This residue amplifier is fully differential with both differential inputs and differential outputs.

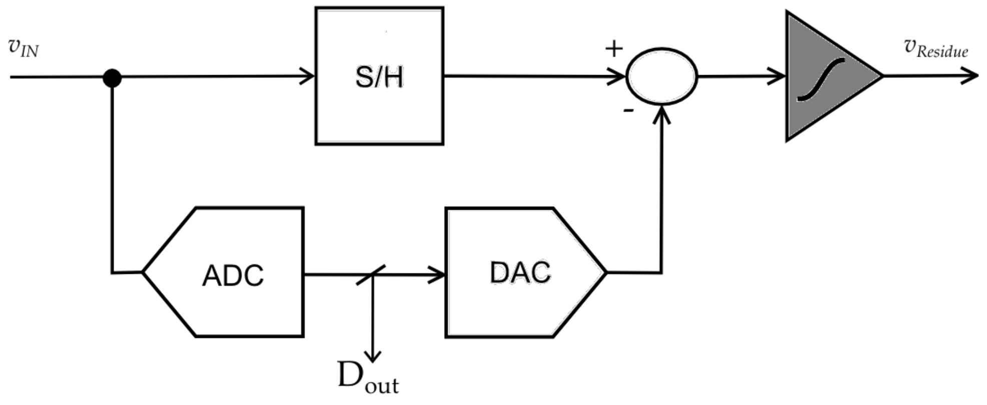


Figure 5.1: System Block Diagram with Highlighted Residue Amplifier

5.2 Design

In the differential amplifier component of this pipeline ADC, an open loop gain stage was used rather than a closed loop. As shown in [8], an open loop amplifier can reduce the power consumption by 50%-70%. The major drawback of the open loop amplifier is that a nonlinear gain is introduced to the residue voltage. This nonlinearity will be corrected using a digital algorithm.

In order to obtain a large bandwidth in each stage of the pipeline ADC, the residue amplifier is designed to have small inter-stage gain. Since only a small gain is needed in each stage, the main differential pair amplifier is designed with resistive loads to reduce the circuit complexity.

The stages of the pipeline ADC, including the residue amplifier, are scaled to reduce power consumption. The residue amplifier for the smallest stage with the load capacitor of 125fF was designed as shown in Figure 5.2.

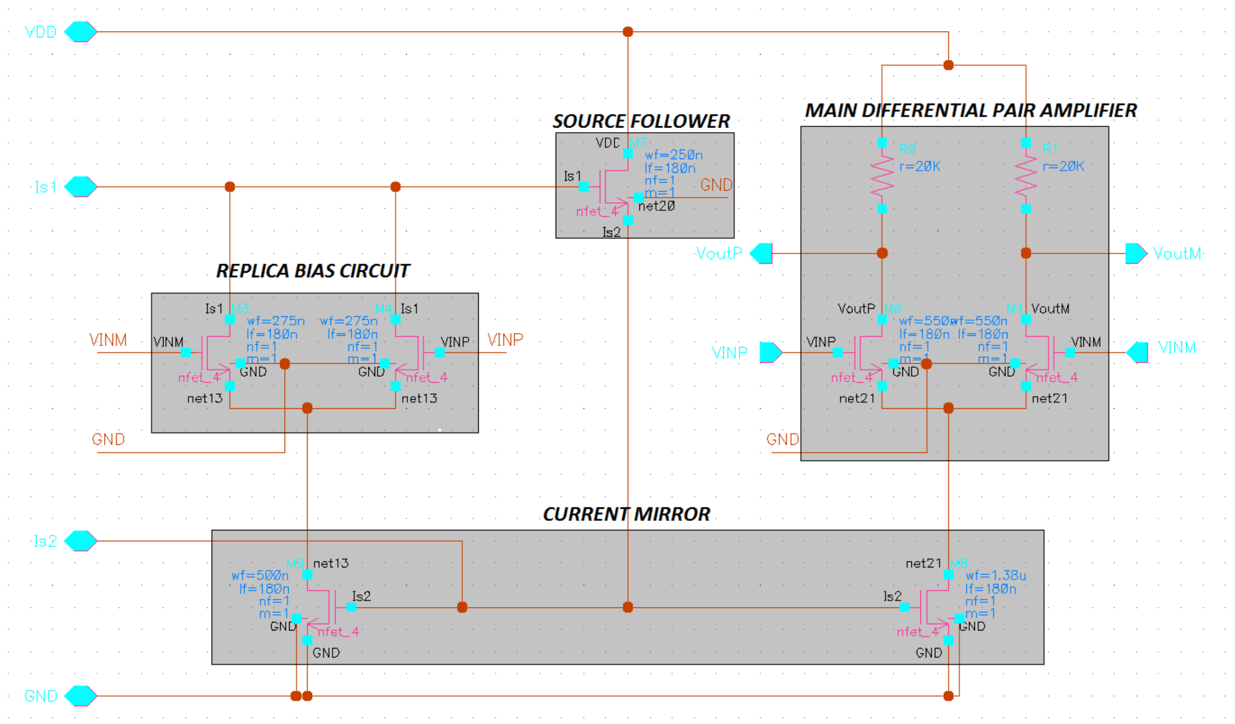


Figure 5.2: Residue Amplifier

5.3 Fundamental Components of the Residue Amplifier

The design of the residue amplifier is composed of the four major blocks listed below:

- A open loop differential pair amplifier
- A replica bias circuit
- A current mirror
- A source follower

5.3.1 Open Loop Differential Pair Amplifier

This residue amplifier provides a gain of 2.4. Theoretically, one bit is output at each stage, the gain of the amplifier should be $2^N = 2$. However, due to the voltage dividers between the capacitors in the switch capacitor network, and the gate capacitance of the MOSFETs in the differential pair amplifier, there is some voltage loss in the residue voltage. To compensate for the loss, the residue amplifier provides a gain of 2.4.

The differential pair amplifier is implemented with a passive load. Its power supply ranges from ground to VDD = 1.8V. Its two differential input signals are in the range of -0.5V to 0.5V with a common mode voltage of 0.9V.

5.3.1.1 Derivations of Resistive Load Values and Bias Current

The important parameters of the differential amplifier are its gain, bandwidth, noise level and settling time. Since the sampling frequency is 20MHz, conversions need to be completed within 50ns. This leaves 25ns for the amplifier to settle.

Since the smallest stage residue amplifier has a load capacitor of 125fF, there must be at least 8.3τ (RC time constants) to output 12bits resolution signal at the load capacitor [9].

$$ln(2^{-numberofbits}) = numberof\tau \tag{5.1}$$

$$ln(2^{-12}) = 8.3\tau \tag{5.2}$$

In order to find the resistance, the value of τ is required. The minimum number of time constants needed is 8.3. To allow for a decent headroom, 10τ was used.

$$\tau = \frac{25ns}{10} = 2.5ns \tag{5.3}$$

Since the value of the RC time constant (2.5ns) and the value of the load capacitor (125fF) are known, the resistance can be found as follows:

$$R_d = \frac{\tau}{C} = \frac{2.5ns}{125fF} = 20k\Omega \tag{5.4}$$

The effective voltage ($V_{eff} = V_{gs} - V_{th}$) is set to 0.25V. The transconductance (g_m) of the MOSFET was determined by the following quantitative analysis:

$$Gain = -g_m * R_d \tag{5.5}$$

$$g_m = \frac{-Gain}{R} = \frac{-(-2.4)}{20,000} = 120\mu A \tag{5.6}$$

Once the transconductance was calculated, the value of the bias current was found:

$$I_d = \frac{g_m * V_{eff}}{2} = 15\mu A \tag{5.7}$$

$$I_{bias} = 2 * I_d = 30\mu A \tag{5.8}$$

With the calculated value of the transconductance of the MOSFETs and the bias current, the differential pair was designed. The requirements of the Jazz $0.18\mu m$ process CMOS technology dictates that the largest allowable width of the MOSFET is 250um. The lengths of the MOSFETs were chosen to be 180nm due to the process size. The widths of the MOSFETs, the transconductance and the bias currents were adjusted to produce the required gain. The functionality was verified by analyzing the DC operating point during DC analysis. The method used to achieve the value of the width of the MOSFETs is to tweak the width. We tweaked the transconductance and the bias current to get the desired values, which was verified by analyzing the DC operating point of the MOSFETs after DC analyses. The width of the MOSFETs were set to 550nm.

5.3.2 Replica Bias Circuit

The purpose of the replica bias circuit is to duplicate the behavior of the differential pair. Since we used a current mirror configuration to bias the differential pair, we have to maintain a consistent current depending on its input voltages that are within the allowed range. The drain to source voltage of the MOSFET at the main current source need to be the same as that of the transistor at the bias current. The replica bias circuit in the residue amplifier keeps the main current source in the saturation region while providing a corresponding consistent bias current to the main differential pair amplifier. The inputs of the replica are connected to the same inputs as the main differential pair circuit. The schematic of the replica bias circuit can be seen in the following Figure 5.3.

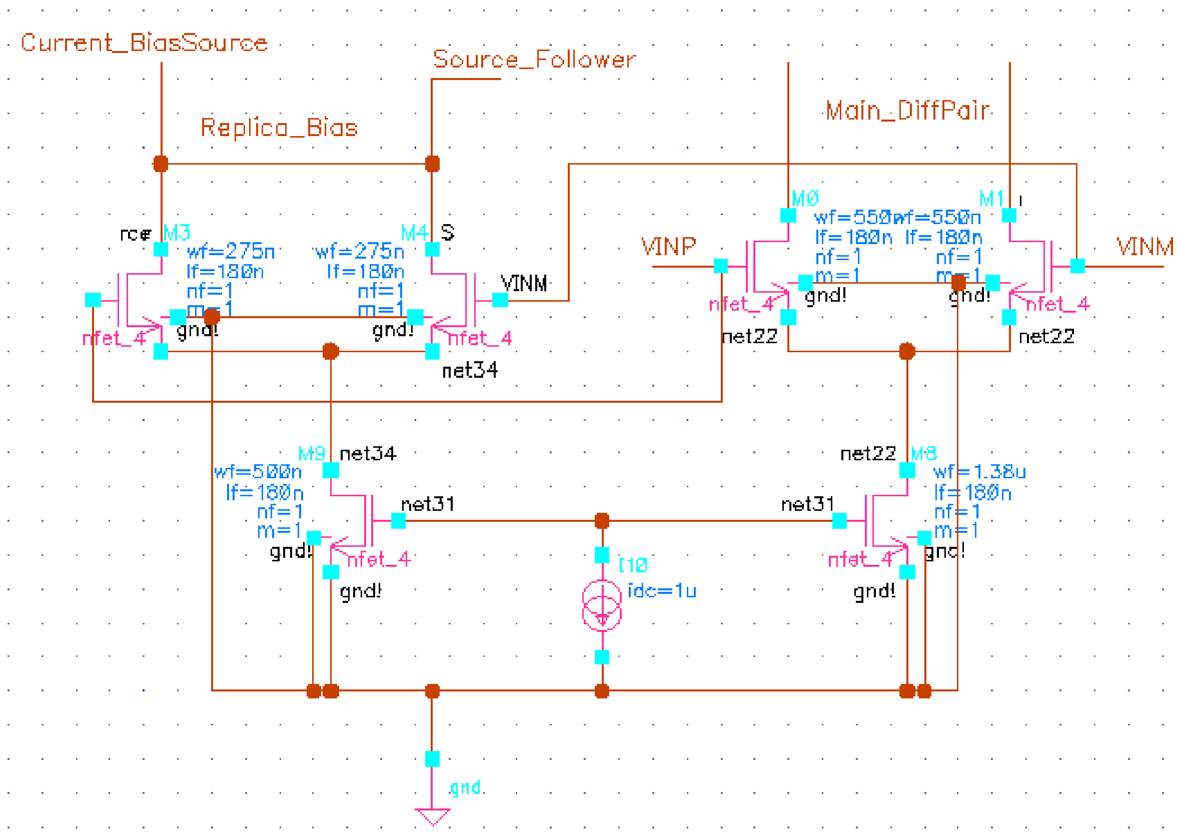


Figure 5.3: Replica Bias Circuit

Because this replica bias is a duplicate of the differential pair, the size of the transistors need to be chosen to correspond with the devices in the main differential pair. Normally, the replica bias would be 10 times smaller than that of the differential pair in order to reduce the power consumption and the die area. In this design, the transistors of the replica bias are half the size of the transistors in the differential pair because the smallest width of a transistor can be in the Jazz 0.18μm process is 220nm. Since the transistors in the main differential pair are 550nm, the smallest width possible replica bias is 275nm, which is half of the width of the differential pair.

5.3.3 Source Follower

A source follower is used in this residue amplifier as a voltage level shifter. It is designed with a N-type MOSFET. Without the source follower, some of the components of the residue amplifier will crash into the triode region. For example, if the replica bias is directly wired to the current from the bias source, there is a high voltage drop across the current source due to its high output impedance. Since the majority of the voltage is across the current source, the voltage between the drain and the source of the replica bias becomes very small. Consequently, the replica bias to crashes into the triode region. Similarly, if the gates of the transistors in the current mirror are directly connected to the supply voltage, then the gate to source voltage becomes very high, which in turn forces the transistors into the triode region. Hence, the source follower keeps the transistors in the replica bias and the current mirrors in the saturation region at any differential input voltage.

5.4 Simulation and Results

Simulations of the residue amplifier were performed to verify the functionality of the residue amplifier within the requirements of the design.

5.4.1 Gain

In order to verify the gain and the output common mode voltage of the residue amplifier, the input DC voltage was swept from -0.5V to 0.5V and the input and output waveforms were analyzed. From the transfer function plot of the differential input to the differential output shown in Figure 5.4, the desired gain of 2.4 was verified.

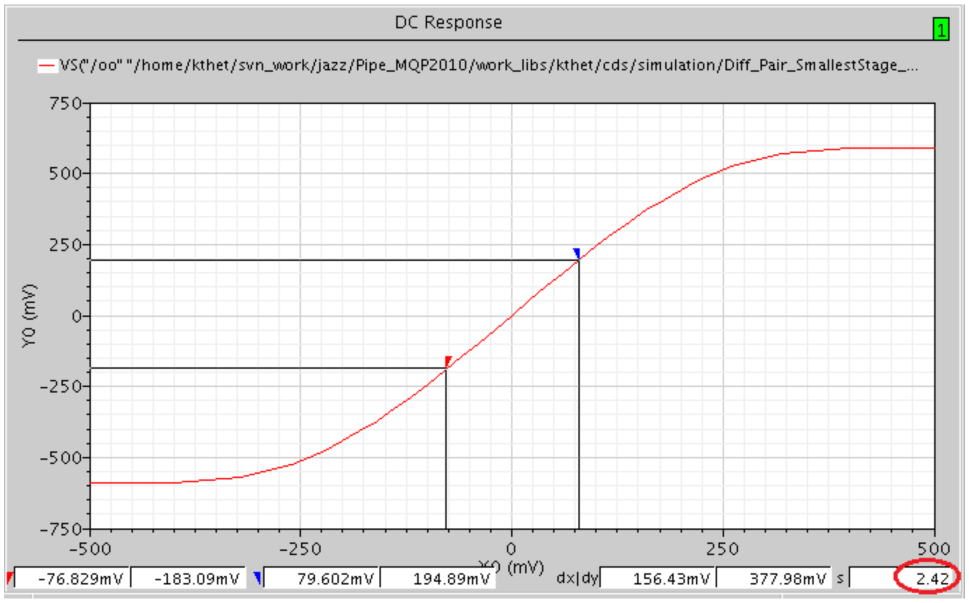


Figure 5.4: Transfer Function Plot of Differential Input to Differential Output

5.4.2 Output Common Mode Voltage

The output common mode voltage of the residue amplifier can be calculated with the equation of $Vo_{CM} = V_{DD} - \frac{I_{Bias}}{2} R_D$. With the supply voltage, V_{DD}, of 1.8V, a bias current of 30μA and a resistor value of 20$k\Omega$, it was observed that the output common mode voltage is equal to 1.5V. To verify this value, the transfer function plot of the differential input to each output is plotted as shown in Figure 5.5. From this, we can observe that the calculated value and the simulation value are the same.

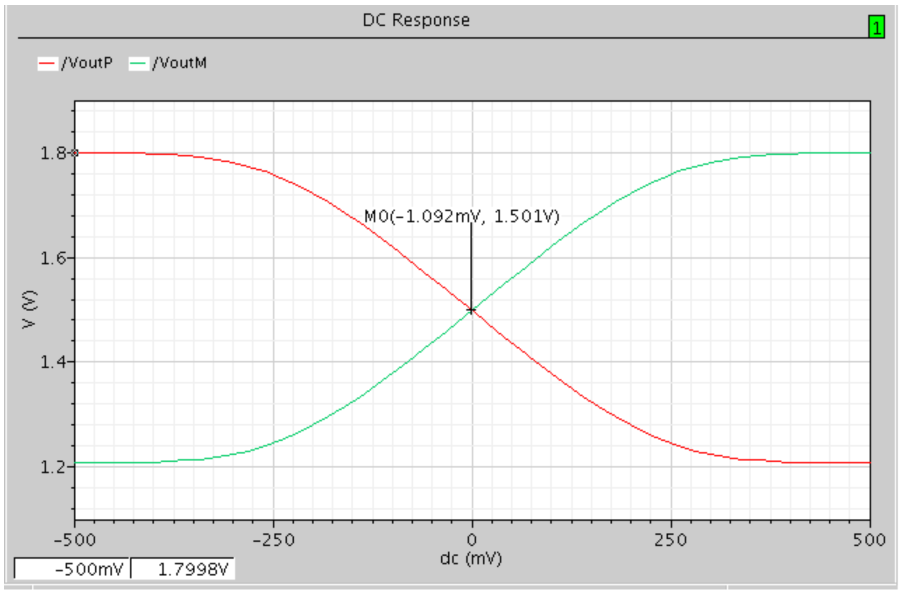


Figure 5.5: Transfer Function Plot of Differential Input to Each Output

5.4.3 Settling Time

When designing a pipeline analog to digital converter, it is important to make sure that each stage processes in its allowed conversion time. The output of each stage has to settle before continuing to the next stage. Since the output of the residue amplifier is the input of the next stage, a transient analysis was performed to investigate the settling time, T_s, at the differential output. The transient analysis of the residue amplifier can be seen in Figure 5.6. From the graph, it was observed that the differential output of the residue amplifier of the smallest stage can settle in about 13ns, which is within the limited conversion time for each sample.

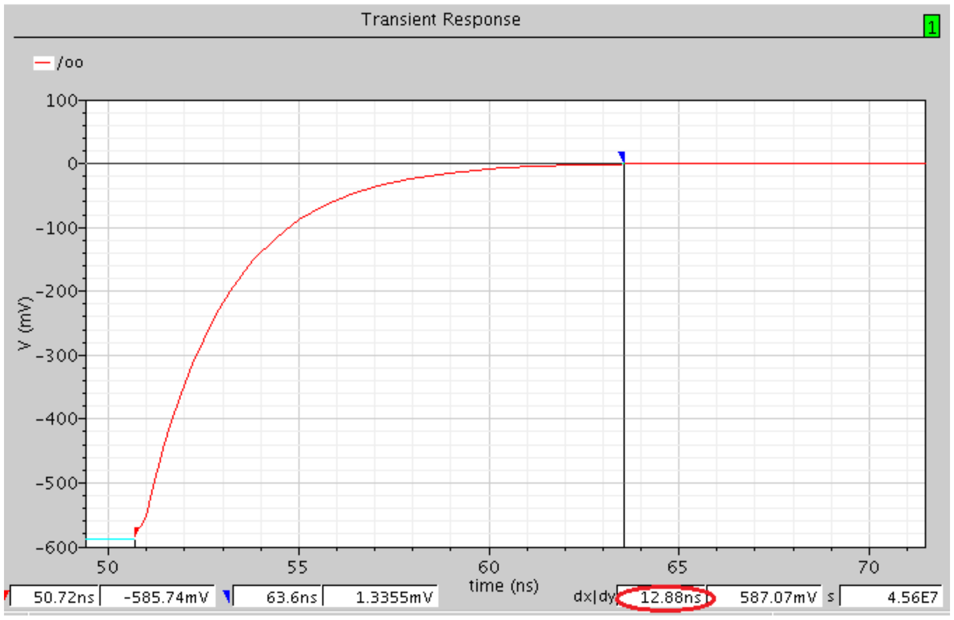


Figure 5.6: Settling Time Plot

5.4.4 Bandwidth

One of the main specifications that needs to be included in every amplifier design is its bandwidth and gain bandwidth product. AC analysis was performed and the frequency response of the residue amplifier was plotted as shown in the Figure5.7.

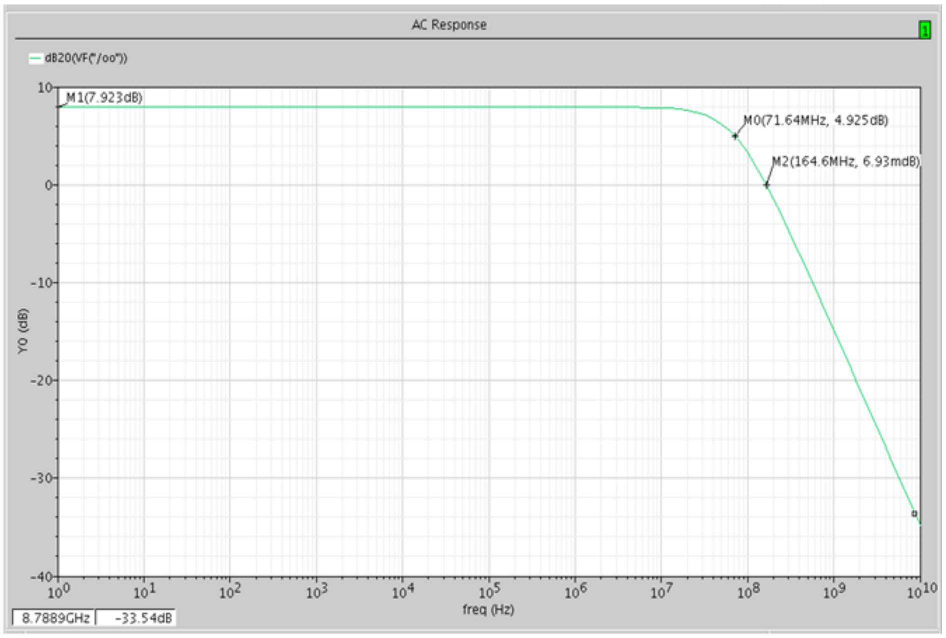


Figure 5.7: Bandwidth of the Residue Amplifier

From Figure 5.7, the DC gain is equal to 7.9dB. When this dB value is converted with the following equation $10^{(\frac{7.9}{20})}\approx 2.4$, the desired gain of 2.4 is acquired. The 3dB frequency equals 71.6MHz and the gain bandwidth product is 164.6MHz.

5.4.5 Noise

The noise of the residue amplifier has to be less than the quantization noise of the whole ADC. In order to verify this, the following noise analysis was performed. The squared output noise plot of the residue amplifier is shown in the Figure 5.8.

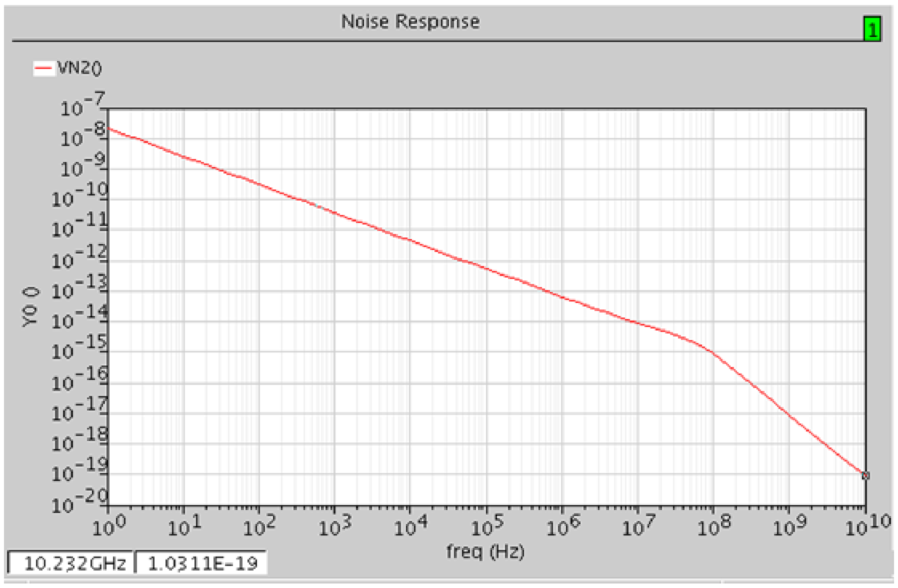


Figure 5.8: Squared Output Noise Plot

From the plot, 1/f noise, also known as "Ficker Noise," is evident and dominant in the residue amplifier. In order to measure the noise, the squared output noise curve was integrated from 1Hz to 71.6MHz (the bandwidth of the residue amplifier). The square root of this value is the noise.

$$\int SquaredOutputNoise = \sigma^2 \tag{5.9}$$

$$\sigma = \sqrt{\sigma^2} = 0.405mV \tag{5.10}$$

In order to reduce the noise, the width and length of the main differential pair were increased by a factor of 10 from 550nm to 5.5μm for the width and 180nm to 1.8μm for the length. Then, the noise analysis was performed and the result is shown in Figure 5.9. By using the equation5.9, it was observed that the noise was $\sigma = 0.34mV$.

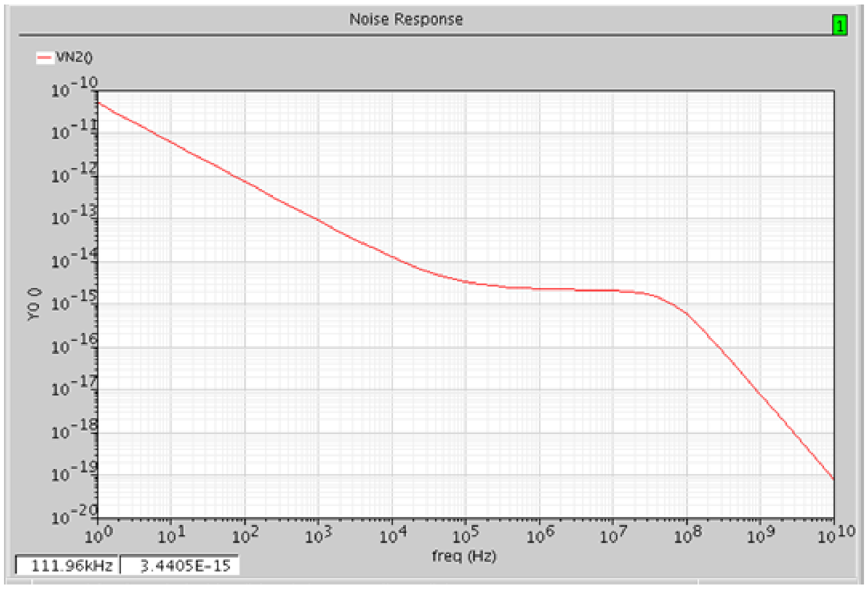


Figure 5.9: Squared Output Noise Plot (Increased the Size of the Main Differential Pair)

Comparison It can be seen that when the size of the main differential pair is increased, the noise of the residue amplifier decreased. Due to an increase in size, however, the amplifier gets a little bit slower because the settling time increases. Another trade off is that the bandwidth of the residue amplifier decreases. The comparisons of bandwidth and noise between the modified residue amplifier and the original design are shown in Figure5.10 and Figure 5.11.

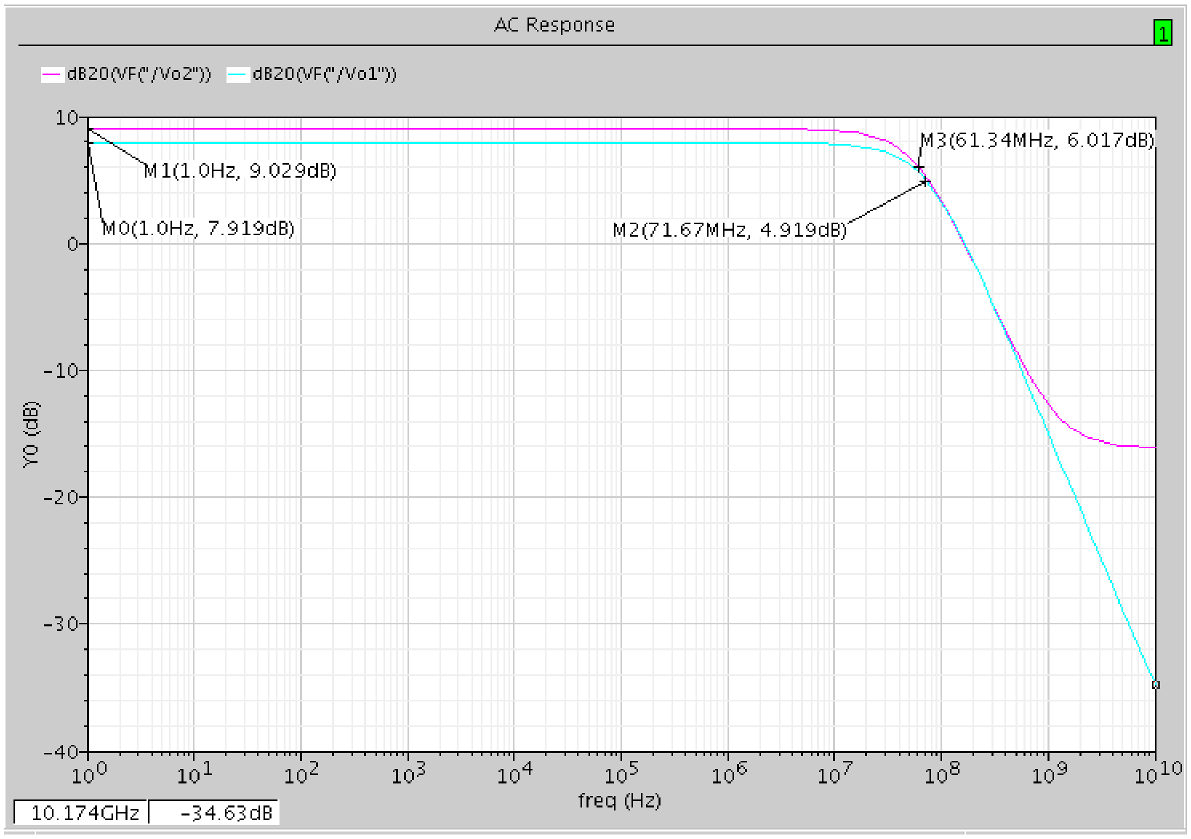


Figure 5.10: Bandwidth Comparison of Design with W/L (blue) and 10W/10L (purple)

In Figure 5.10, the purple curve represents the bandwidth of the residue amplifier design with 10W/10L and the blue curve represents the bandwidth of the residue amplifier design with (W/L). It can be observed that the purple curve has a bandwidth of 61.3MHz which is smaller than the bandwidth of the blue curve which is 71.67MHz. The residue amplifier with a larger width and length design has a smaller Ficker noise than the residue amplifier with the smaller size.

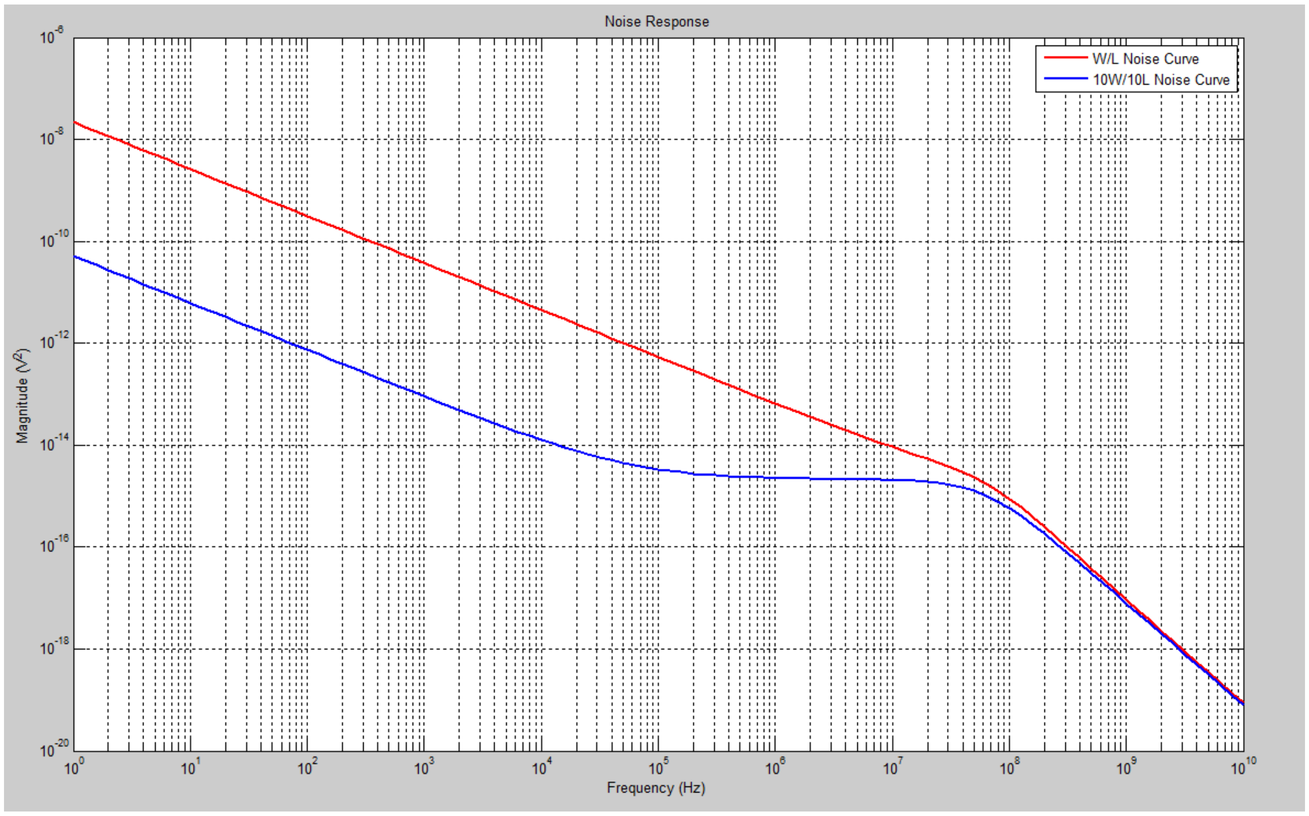


Figure 5.11: Bandwidth Comparison of Design with W/L (red) and 10W/10L (blue)

5.5 Complete Residue Amplifier

Once the design of the residue amplifier was finished, its symbol was created. The following Figure 5.12represents the symbol of the residue amplifier.

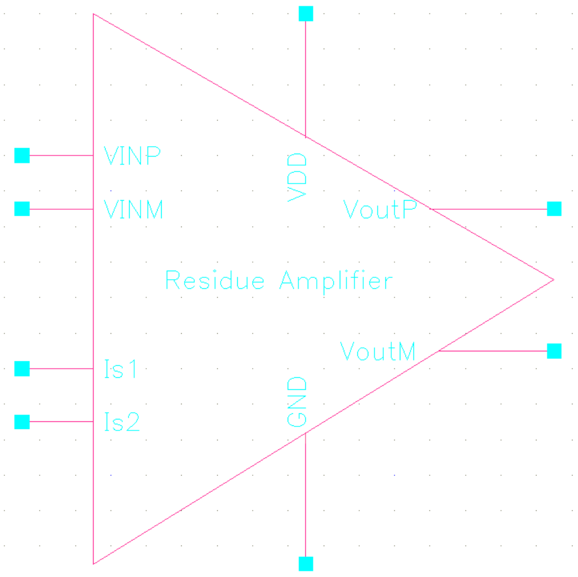


Figure 5.12: Symbol of Residue Amplifier

5.6 Layout

The layout of the smallest stage was designed to fit the stage scaling. Several of the smallest stages can be connected in parallel to produce the corresponding larger stages. The layout design of the residue amplifier was implemented using the common centroid method in the main differential pair to reduce device mismatches. For the replica bias circuit, we could not use the common centroid method due to its very small size. The layout design of resistors was done using an interdigitation pattern of "ABABABAB". The layout design of the residue amplifier can be seen in Figure 5.13. After finishing the layout design, an extracted simulation was performed based on the values extracted from the layout. The residue amplifier layout meets the desired specifications. specifications.

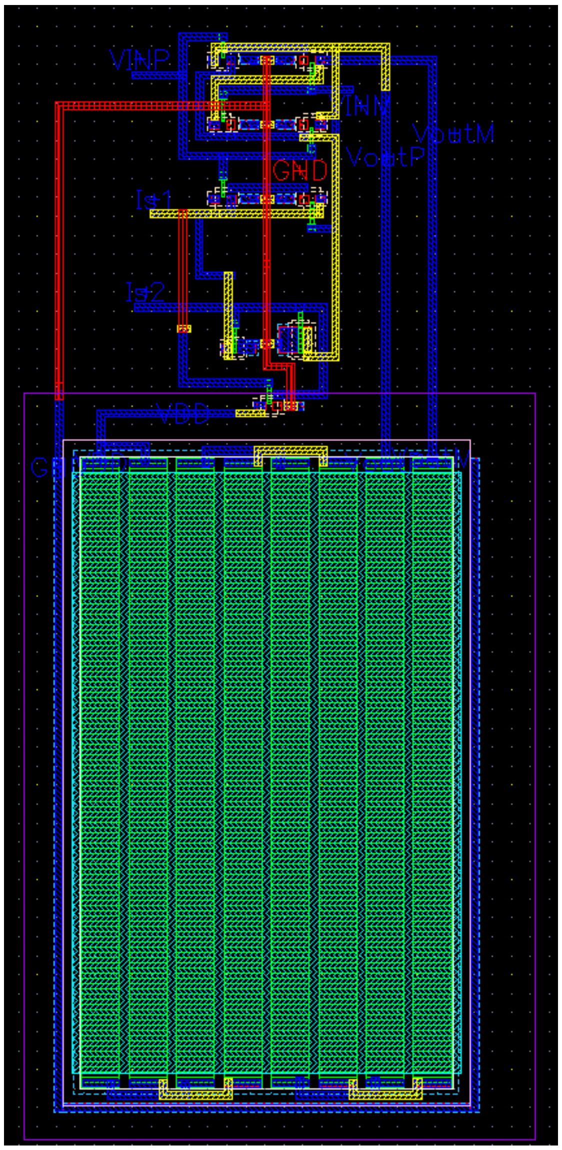


Figure 5.13: Layout of Residue Amplifier

Chapter 6

Sub-ADC

The sub-ADC block determines the digital output of each stage and generates the decision signals that feed into the switched capacitor network decision switches. Two comparators and digital logic comprise the sub-ADC as shown in Figure 6.1.

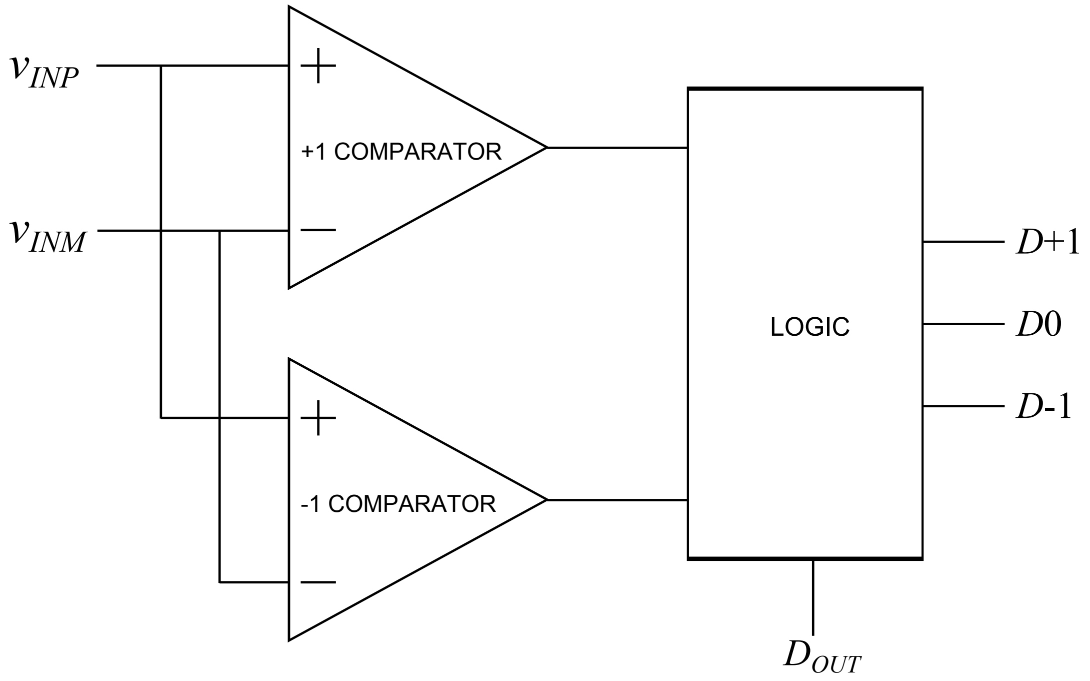
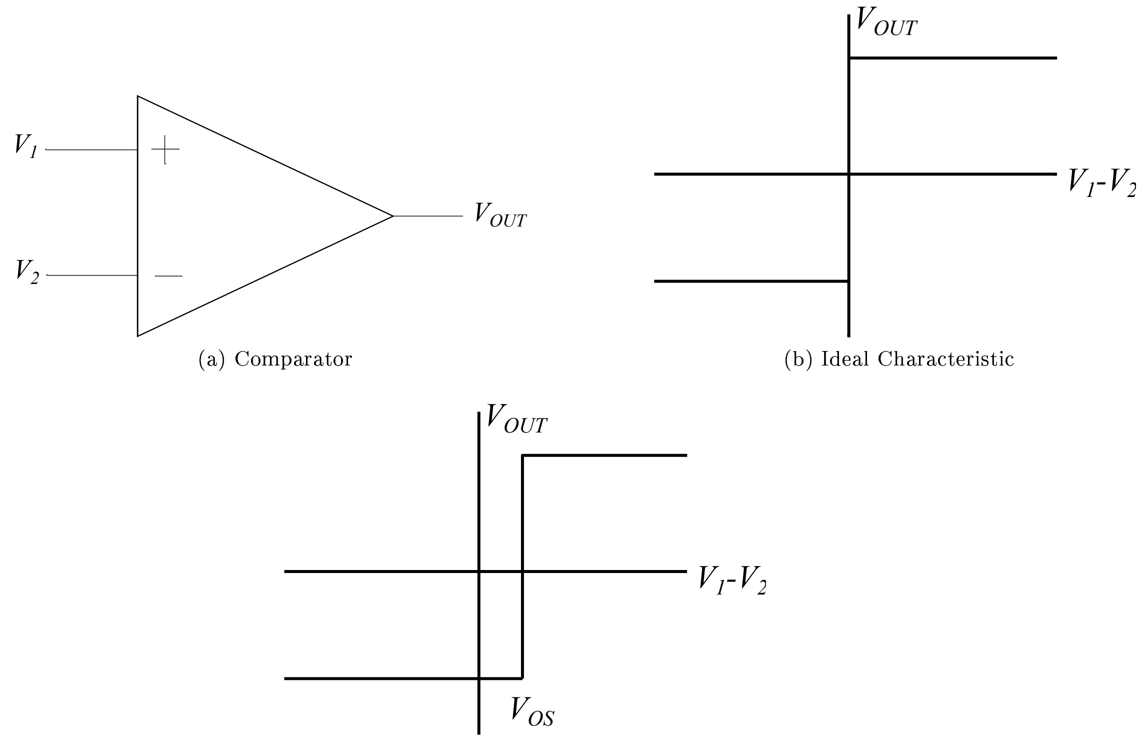


Figure 6.1: Sub-ADC Block Diagram

6.1 Comparator

The most basic ideal comparator, shown in Figure 6.2a, compares two voltage signals and the output indicates whether V_1 is greater than V_2. The input-output characteristic of an ideal comparator can be seen in Figure 6.2b.



(a) Comparator

(b) Ideal Characteristic

(c) Characteristic with Offset

Figure 6.2: Comparator

In order to generate the correct decision signals, the comparators used in the sub-ADC need to have an offset. This offset will shift the characteristic as shown in Figure 6.2c. This means that the decision threshold will be shifted from zero to $V_1 - V_2 = V_{os}$. The desired offsets are $\pm\frac{V_{REF}}{4}$, where $\pm V_{REF} = \pm 0.5V$. The decision for which the difference between the two input signals $(V_{in+} - V_{in-}) > +0.125$ is the "+1" decision in the switched capacitor network; if $(V_{in+} - V_{in-}) < -0.125$, this is the "-1" decision for the switched capacitor network. Any difference that falls between ± 0.125 is the "0" decision. Therefore the "+1" and "-1" comparators were designed to have offsets of $+0.125V$ and $-0.125V$, respectively.

The comparator is comprised of a preamplifier and an analog latch as shown in Figure 6.3.

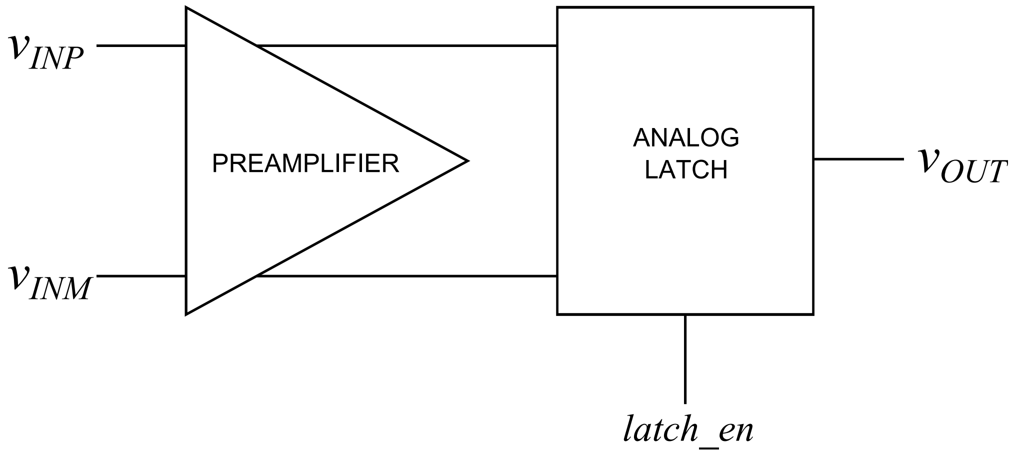


Figure 6.3: Comparator Block Diagram

6.1.1 Preamplifier

The preamplifier circuit, shown in Figure 6.5, is used to create the desired offset in the comparator and amplify the difference between the input signals. The preamp takes its input from the output of the open loop amplifier so the input common mode of the preamplifier is 1.5V. The output common mode of the preamplifier needs to be high enough to turn on the input switches of the analog latch in the next stage. The differential pair has a self-biasing load in which the MOSFETs were sized to maintain a high enough output common mode to turn on the input switches in the analog latch. The self-biased load is also mismatched with a 0.8:1 ratio to ensure that the feedback in the self-bias only has one stable point.

The necessary offset is created by mismatching the widths of the MOSFETs in the common source amplifiers and the differential pair. The common source amplifiers and the differential pair were mismatched such that each created half of the total required offset. Figure 6.4 shows the the results of a DC sweep of the input voltage and shows the offsets created by both the common source amplifiers and the differential pair mismatches.

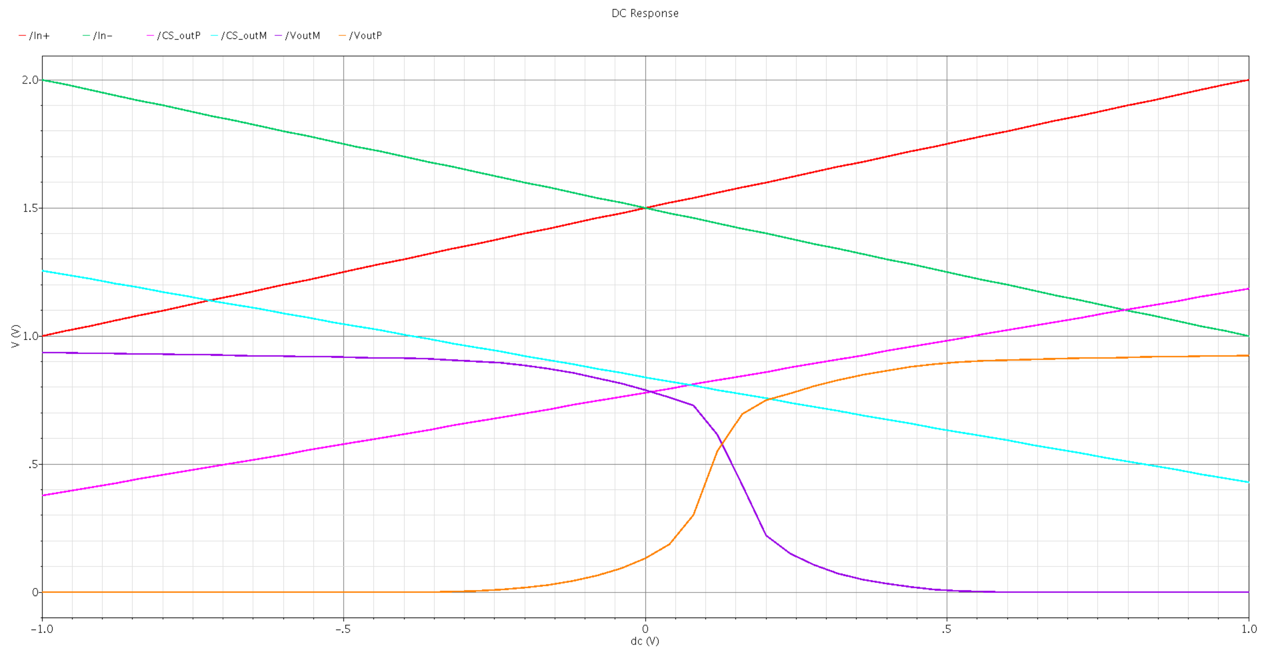


Figure 6.4: DC Analysis

Feedback from a replica circuit is used to bias both the common source amplifiers and the differential pair. The replica circuit allows the preamplifier design to withstand manufacturing variations and still function correctly. The inputs to the replica circuit are set to be exactly at the decision level: $V_{inP\_rep} = V_{incm} + \frac{V_{REF}}{8} = 1.5625V$ and $V_{inM\_rep} = V_{incm} - \frac{V_{REF}}{8} = 1.3475V$. This choice will yield $V_{inP\_rep} - V_{inM\_rep} = 1.5625V - 1.3475V = 0.125V$, i.e. the decision level transition voltage. Feedback from the replica circuit controls the bias currents in the common source amplifiers and differential pair in the main circuit.

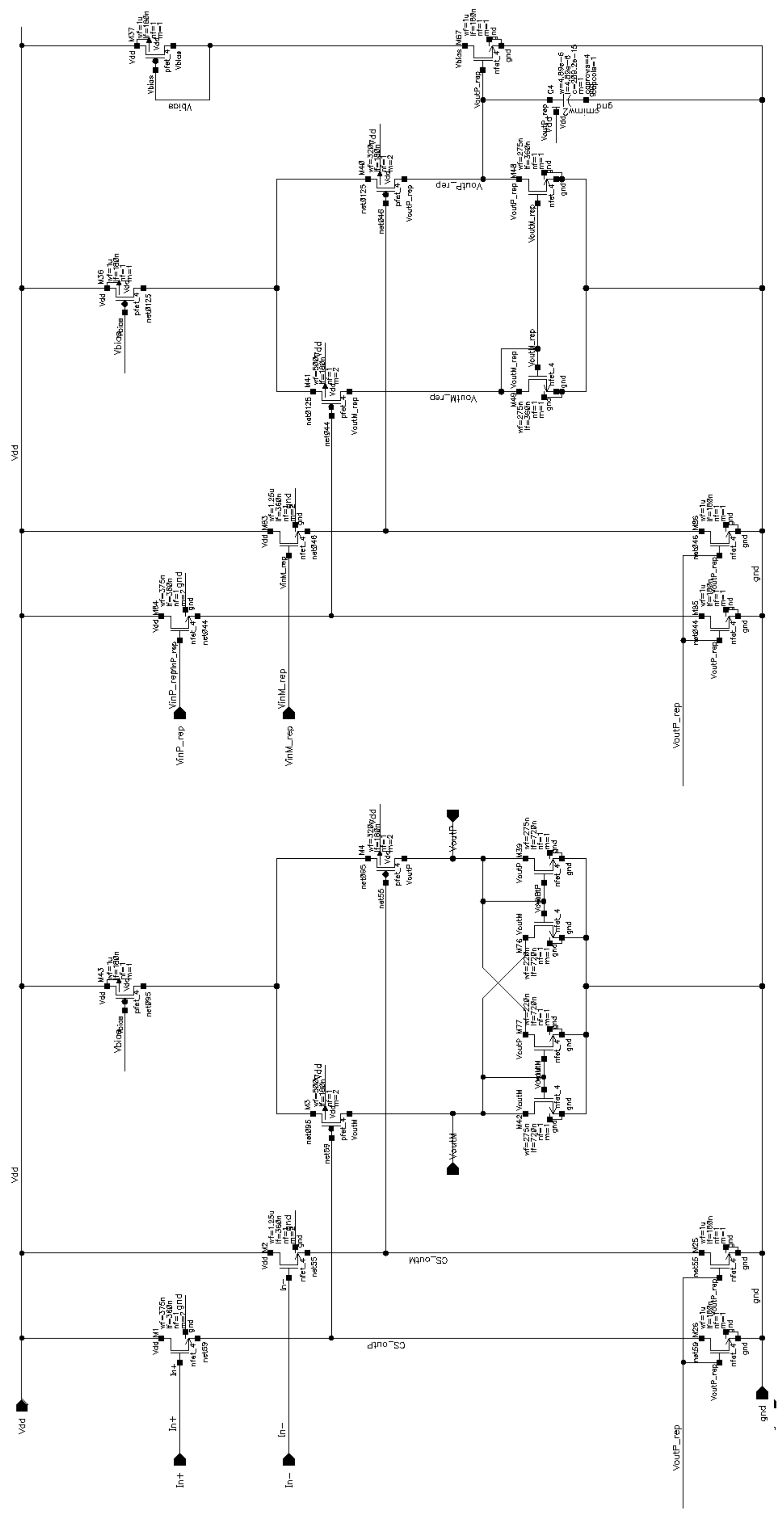


Figure 6.5: Preamplifier Circuit

The layout for the preamplifier was completed using common centroiding in the common source amplifiers and differential pairs in both the main circuit and the replica. The main circuit and preamplifier layout were arranged to mirror one another and close together so that the manufacturing variations in the main circuit and replica circuit would be as similar and correlated as possible to increase the effectiveness of the replica circuit feedback.

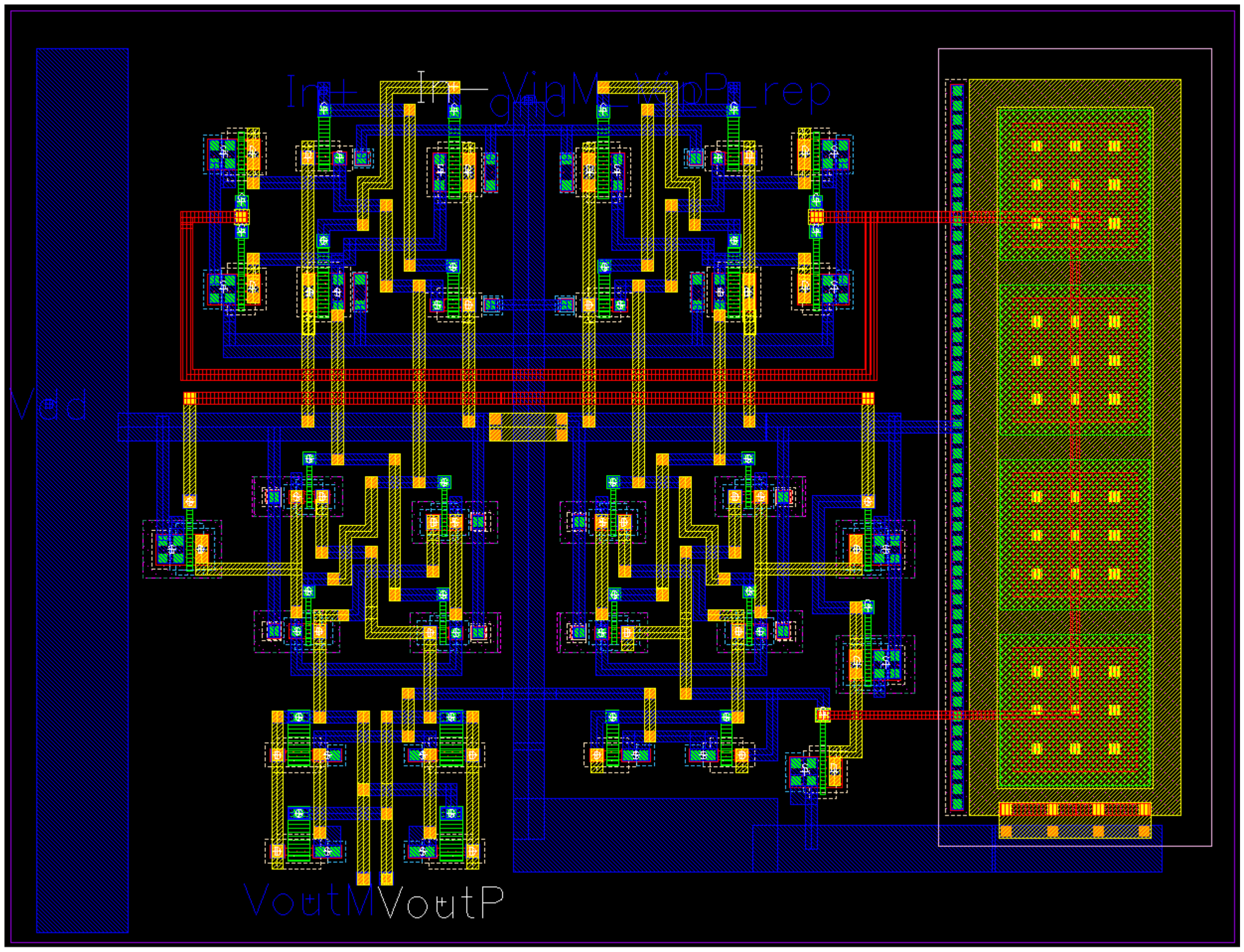
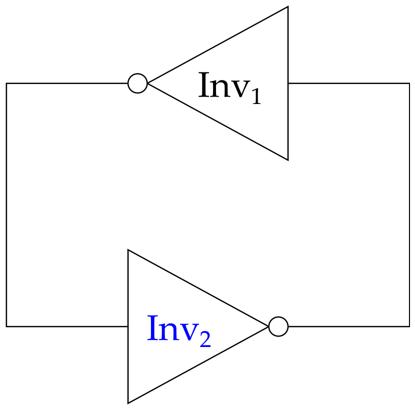


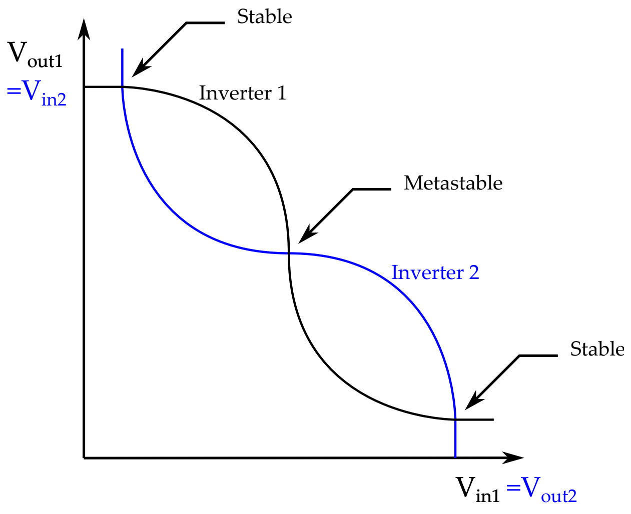
Figure 6.6: Preamplifier Layout

6.1.2 Analog Latch

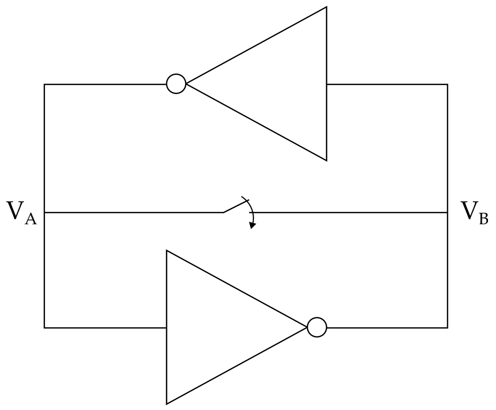
The analog latch drives its output to one rail or the other depending on the input it receives from the preamplifier. The basic latch is two connected inverters as shown in Figure 6.7a. This configuration will have the input output characteristic shown in Figure 6.7b. The points of stability and metastability occur at the intersection of the input-output curves of the two inverters. Figure 6.7b shows two stable points, one high (V_H) and one low (V_L), and a metastable point in the middle V_{Meta}. The difference between stability and metastability in a latch can be visualized like a ball balanced at the peak of a hill versus a ball at the foot of a hill. The ball at the foot of the hill needs to be moved up the hill to reach another point of stability or metastability, while the ball balanced on the peak requires only a small push to fall to a stable point. Likewise, to move from $V_L$to V_H or vice versa, a significant change must be applied, however, only a small voltage variation is required to cause the metastable point to "fall" to one stable point or another. If $V_A$and V_B are shorted together with a switch as shown in Figure 6.7c, then the two voltages will both be pulled to the metastable point. Input switches are used to apply voltages that create an imbalance that will push the latch into one of its stable states. Once this imbalance is established, the switch shorting the latch into its metastable state is released and the latch output is driven either high or low.



(a) Basic Latch



(b) Latch Metastability



(c) Latch with Switch

Figure 6.7: Latch

The implementation of these cross connected inverters is shown in Figure 6.8. The *latch_en* signal triggers the latch to stop shorting the inverters together and triggers a delayed signal that disables the inputs to the latch once the latch has been sent to the rail. The outputs of the latch are connected to tapered buffers so that the analog latch can drive the following digital logic gates.

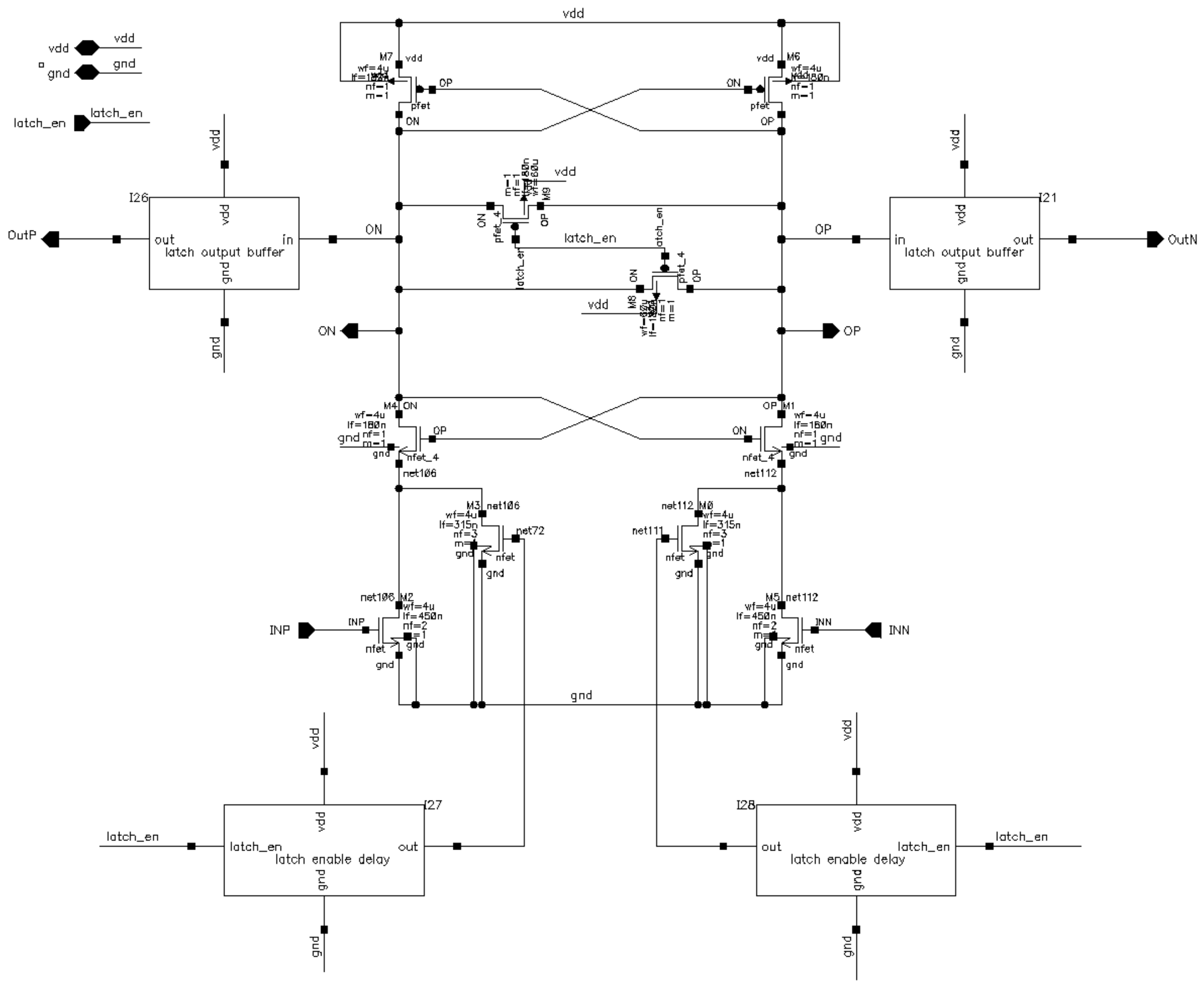


Figure 6.8: Analog Latch Schematic

The functionality of the analog latch was tested and it was verified that the output is latched to the correct rails when *latch_en* is high for input voltages as small as $1\mu V$, shown in Figure 6.9. In the simulation results shown in Figure 6.9, the input switches from a positive differential signal to a negative signal at $0.5\mu s$. For a $1mV$ signal, approximately $1\mu s$ of settling time is required and for a $1uV$ signal, approximately $1.6\mu s$ is needed.

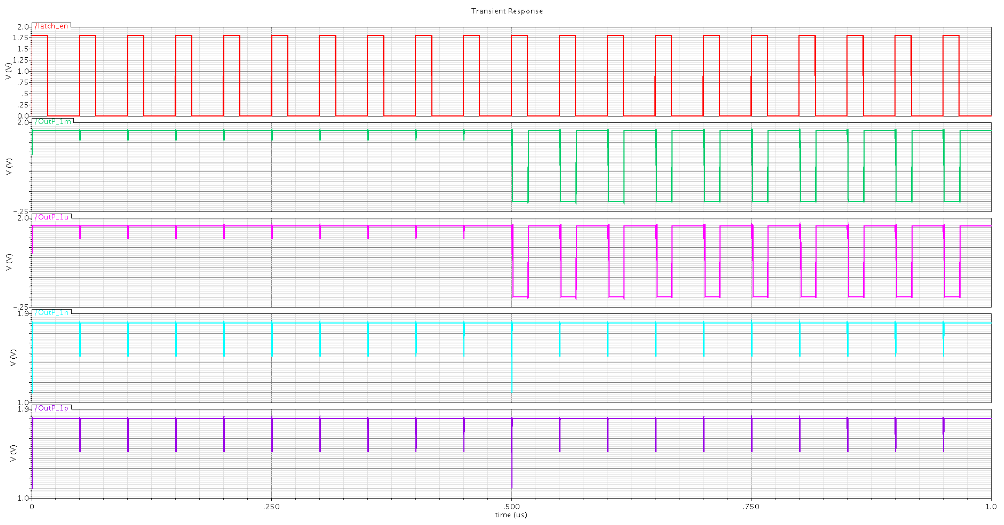


Figure 6.9: Analog Latch Transient Simulation

6.1.3 Complete Comparator

The preamplifier and analog latch were combined to generate the complete comparator circuit. The schematic and circuit symbol for the +1 preamp are shown in Figures 6.8 and 6.11.

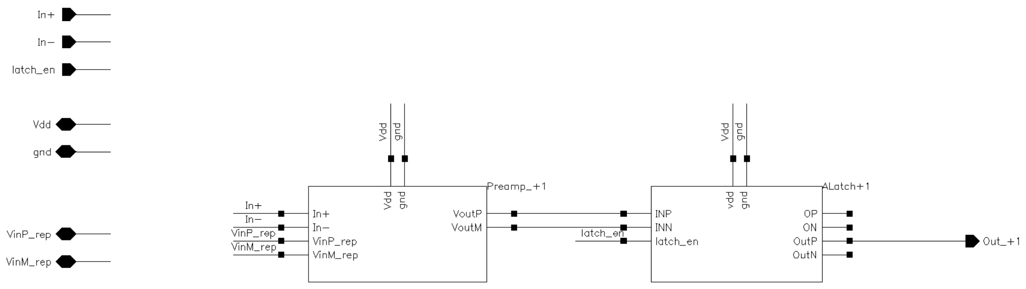


Figure 6.10: +1 Comparator Schematic

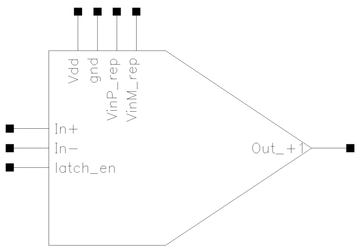


Figure 6.11: +1 Comparator Symbol

6.2 Digital Latch and Logic

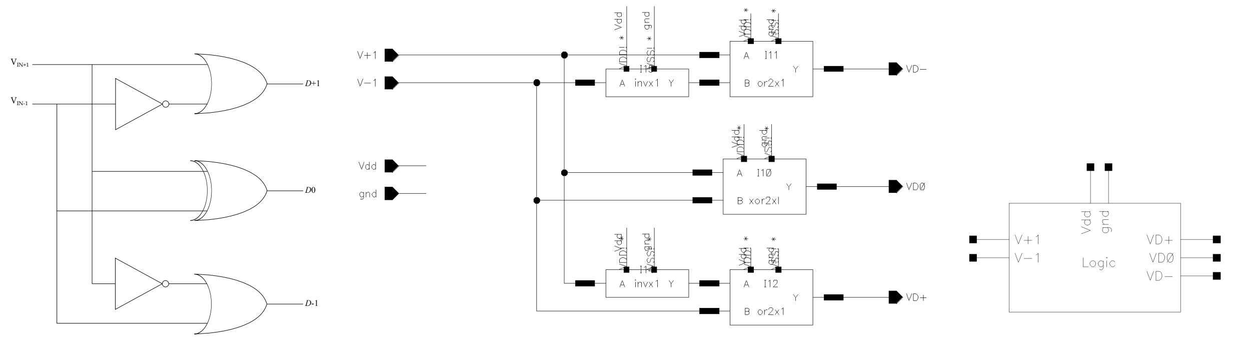
Because the switched capacitor network needs to sample the output of the differential pair and add the decisions from the comparators after the analog latch needs to be disabled, a digital latch was needed to hold the comparator decisions. A D-flip-flop from the Jazz Semiconductors digital libraries was added to the previous circuits to create the complete comparator circuit.

The logic block translates the raw comparator decisions into the signals fed back into the switched capacitor network and provides the ability to enable or disable the decision signals. The logic used to generate the signals for the switched capacitor network is shown in Table 6.1. Because the DAC switches are PMOS, the signal needed to turn on the switch is a zero. The logic required for the output was determined from the logic table.

+1 Comparator	-1 Comparator	D+	D0	D-
0	0	1	0	1
0	1	1	1	0
1	0	0	1	1
1	1	1	0	1

Table 6.1: Combinational Logic for DAC Decisions

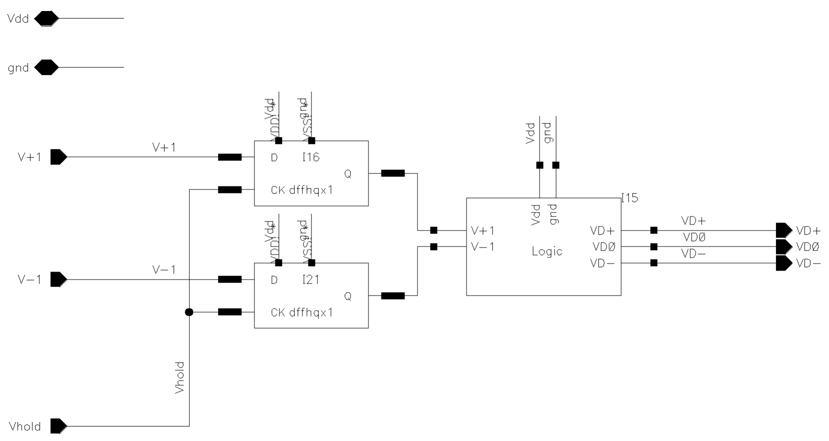
The logic was implemented using discrete components from the Jazz digital library. Figure 6.12a shows the logic required based on Table 6.1 and Figure 6.12b shows the implementation of the logic using the Jazz digital library. The symbol shown in Figure 6.12c was generated for the combinational logic.



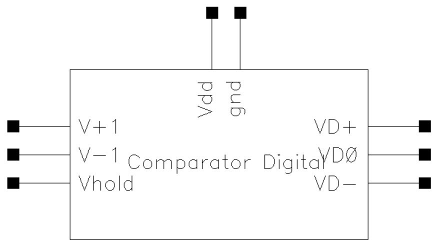
(a) Ideal Logic (b) Logic Schematic (c) Logic Symbol

Figure 6.12: Combinational Logic

In order to latch the output from the analog latch until needed, two D-flip-flops were added to the logic. This allows the comparator decision to be latched until needed by the rising edge of a signal. The addition of the latched to the logic is shown in Figure 6.13a.



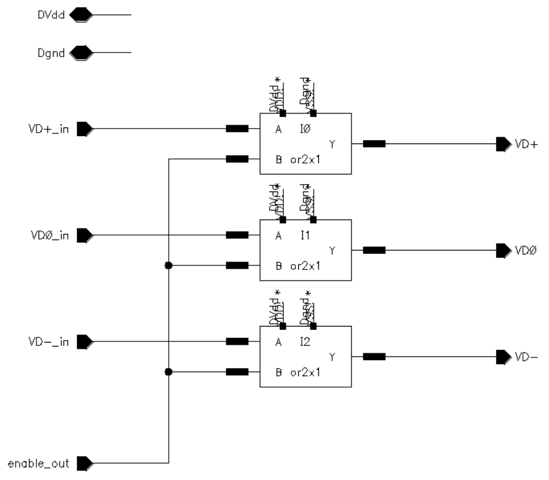
(a) Logic with Digital Latches Schematic



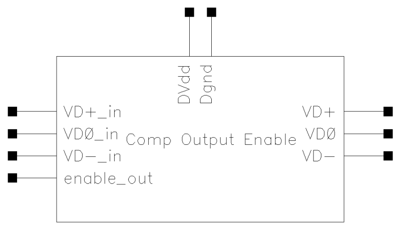
(b) Logic with Digital Latches Symbol

Figure 6.13: Logic with Digital Latches

During the time that the switched capacitor circuit is sampling, the DAC switches should all be off. OR gates were added to the outputs of the logic to allow the output of the logic to be enabled or disabled. The output enabling circuit is shown in Figure 6.14. The outputs are enabled with a logic low and disabled with a logic high. A logic high will generate outputs all high, turning off the DAC switches. With a logic low, the DAC switches will be turned on or off according the the signals generated by the logic.



(a) Comparator Output Enable Schematic

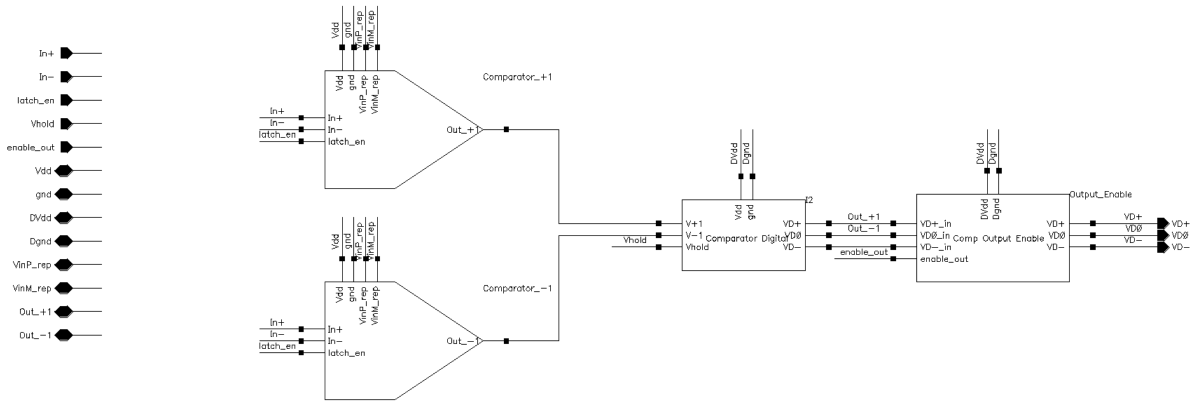


(b) Comparator Output Enable Symbol

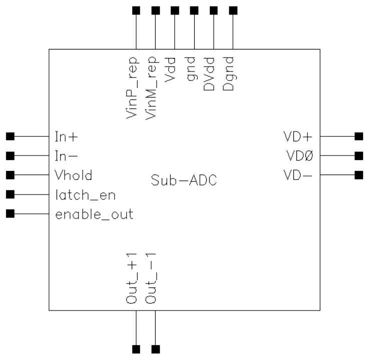
Figure 6.14: Comparator Output Enable

6.3 Complete Sub-ADC Block

The comparators and logic were combined to form the complete sub-ADC block. The schematic is shown in Figure 6.15a. The symbol generated for the sub-ADC is shown in Figure 6.15b.



(a) Sub-ADC Block Schematic



(b) Sub-ADC Block Symbol

Figure 6.15: Sub-ADC Block

Chapter 7

2-bit Flash ADC

The final stage of the pipeline ADC is a 2-bit flash. This 2-bit flash resolves the last two bits of the ADC since the 1.5-bit stage calibration cannot be used on the final stage because there is no following stage. The 2-bit flash is comprised of three comparators at $\pm\frac{V_{REF}}{2}$ and $0V$ and D-flip-flops to latch the outputs.

7.1 Comparators

The comparator design approach for the 2-bit flash was essentially the same as for the sub-ADC. A preamplifier and an analog latch comprise each comparator. The preamplifiers were modified for the new offset requirements. The analog latch used is the same as that used for the sub-ADC.

7.1.1 Preamplifier: ± 1

The ± 1 preamplifiers were designed with the same approach as those used in the sub-ADC. Since the comparison threshold is different, the device width ratios were adjusted to generate an offset of $\pm\frac{V_{REF}}{2}$ instead of $\pm\frac{V_{REF}}{4}$. The reference voltages used in the feedback circuit were also different. The new values were$V_{inP_rep} = V_{incm} + \frac{V_{REF}}{4} = 1.625V$ and $V_{inM\_rep} = V_{incm} - \frac{V_{REF}}{4} = 1.475V$. This yields $V_{inP\_rep} - V_{inM\_rep} = 1.625V - 1.475V = 0.25V$. The 2-bit flash preamplifier circuit is shown in Figure 7.1.

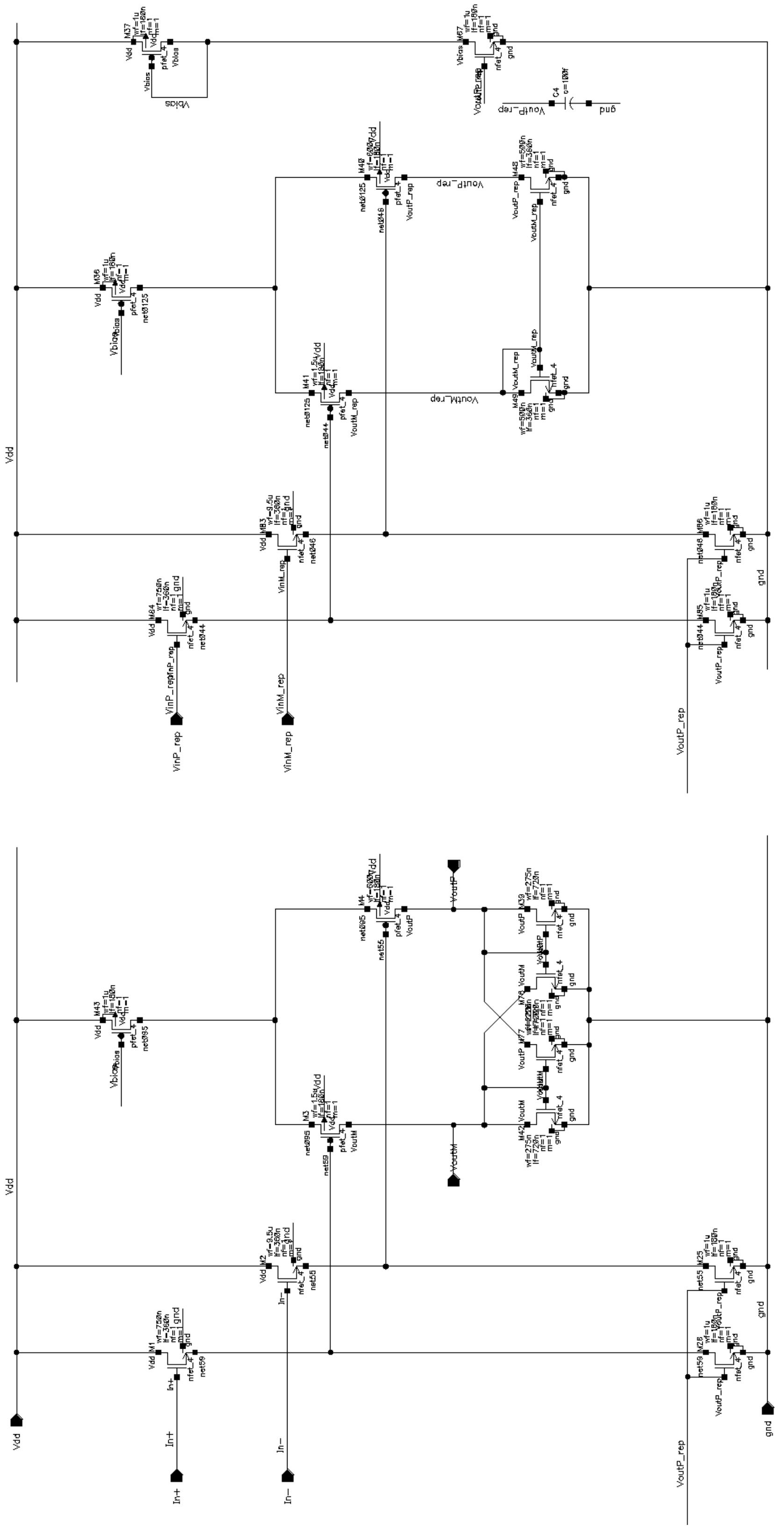


Figure 7.1: +1 Preamplifier for 2-bit Flash

The $\frac{V_{REF}}{2}$ offset can be seen in the simulation results shown in Figure 7.2.

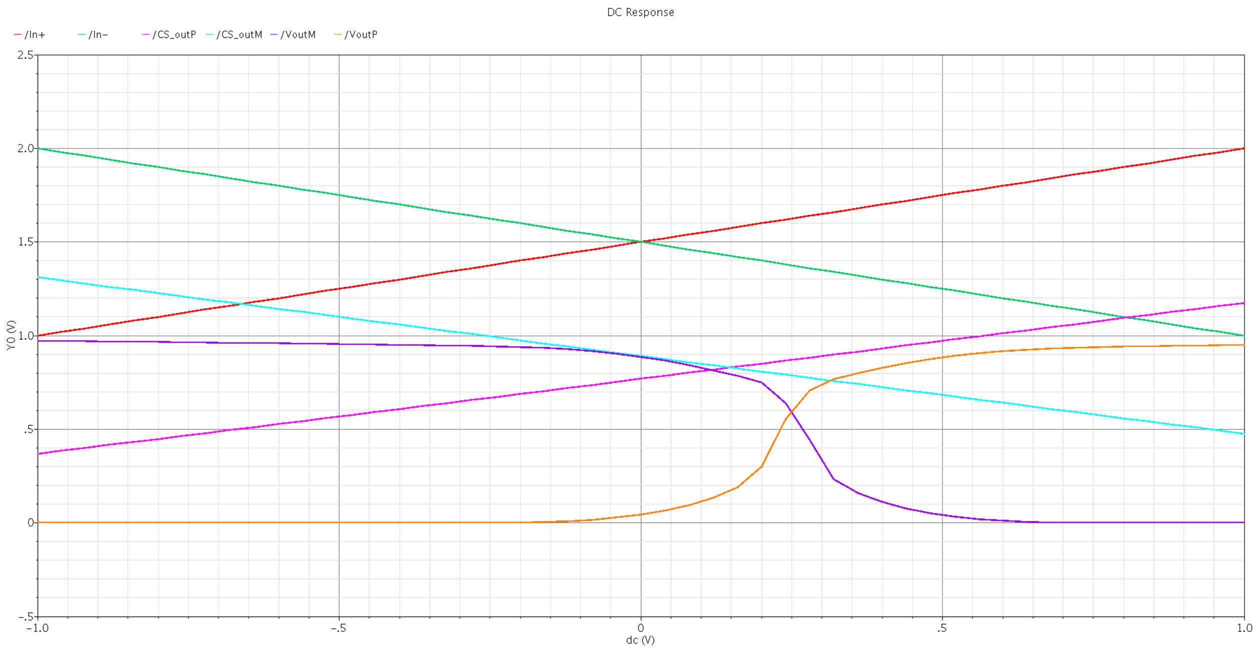


Figure 7.2: DC Analysis of +1 Preamplifier for 2-bit Flash

7.1.2 Preamplifier: 0

The zero preamplifier compares the two inputs directly. Since no offset was needed, the devices in the common source amplifiers and the differential pair were matched. The reference voltage inputs to the feedback circuit were shorted together since desired threshold voltage is $0V$. The feedback circuit was also modified to have an external current bias in the replica circuit because it was found that without an external current source, the circuit would stabilize at a current value that did not generate a sufficiently high output common mode to turn on the input transistors of the analog latch. Figure 7.3 shows the 0 preamplifier circuit.

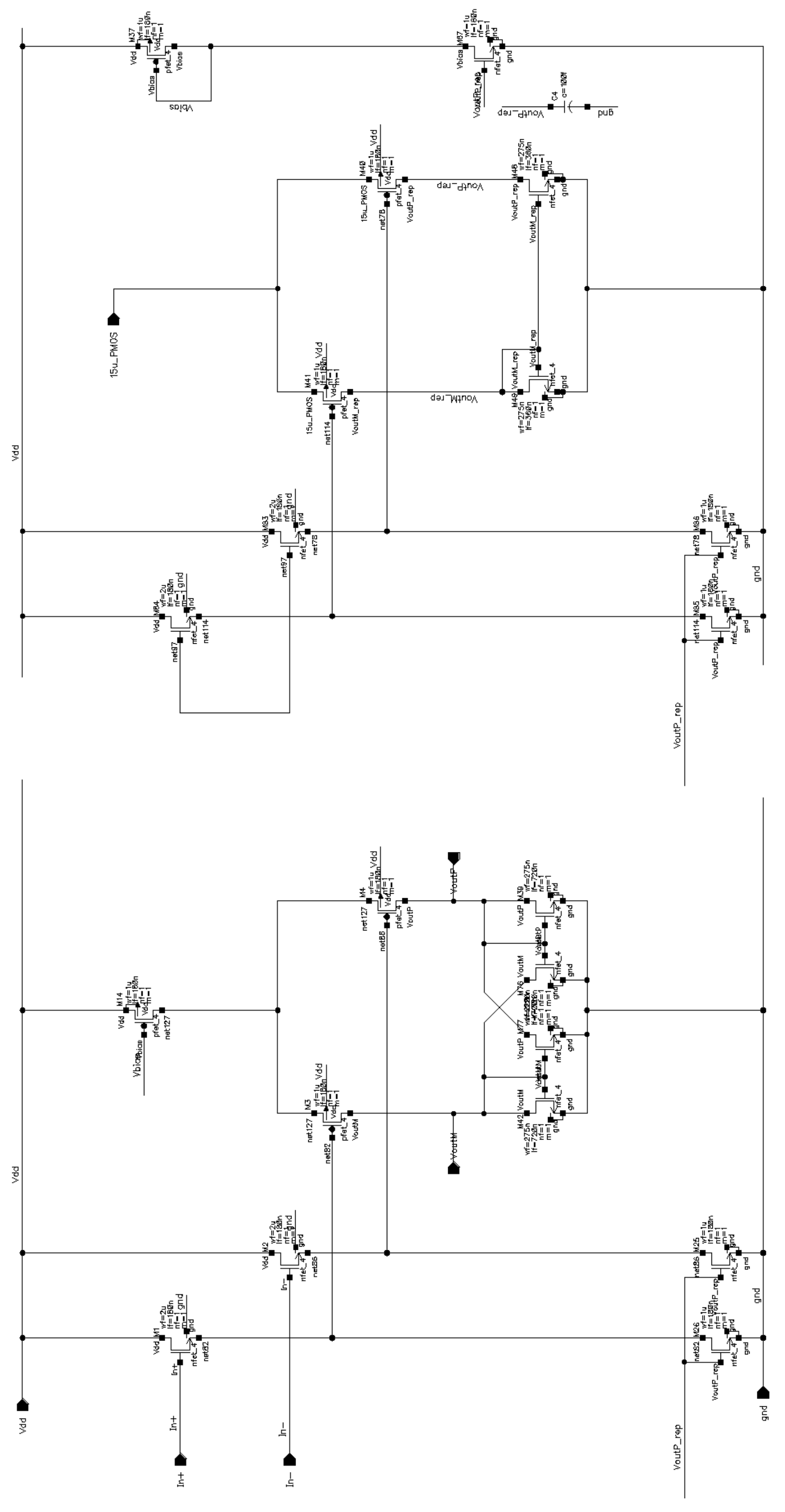


Figure 7.3: 0 Preamplifier for 2-bit Flash

The zero offset can be seen in Figure 7.4.

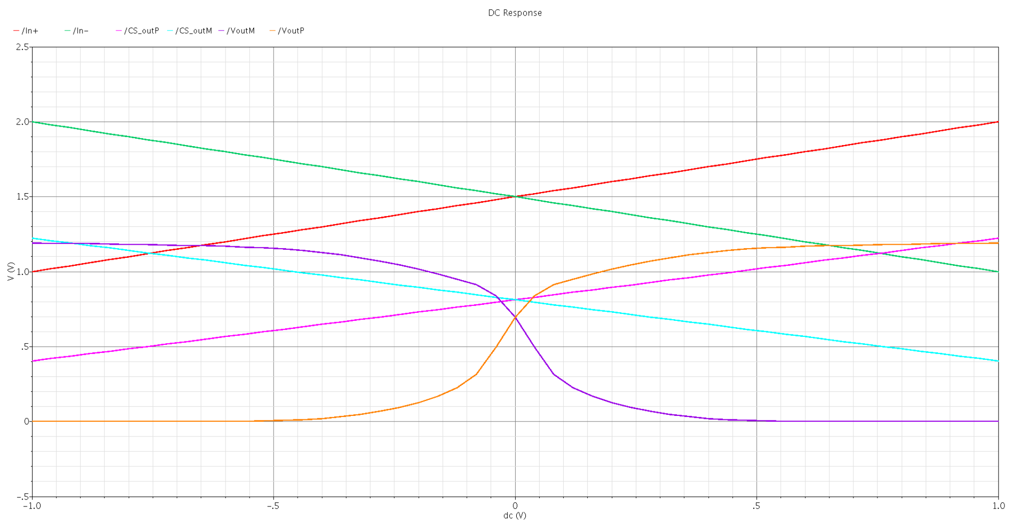
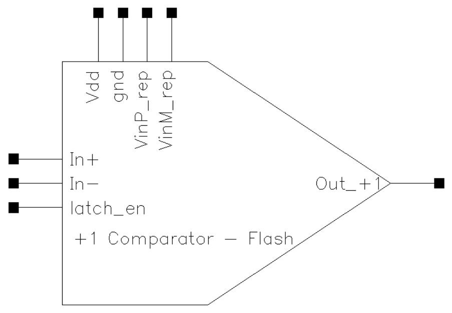


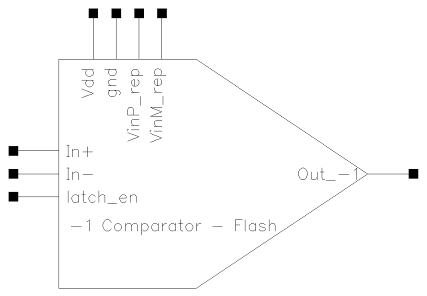
Figure 7.4: DC Analysis of 0 Preamplifier for 2-bit Flash

7.1.3 Complete Comparators

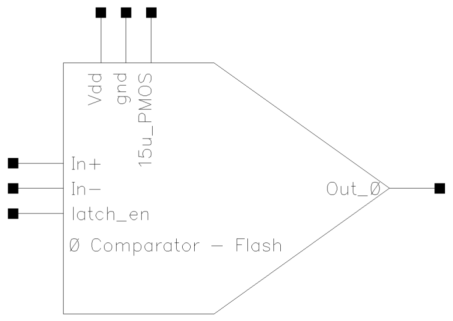
The redesigned preamplifiers were combined with the analog latch used in the sub-ADC as shown in 6.3. Symbols were generated for the +1, 0 and -1 comparators and are shown in Figure 7.5.



(a) +1 Comparator



(b) -1 Comparator

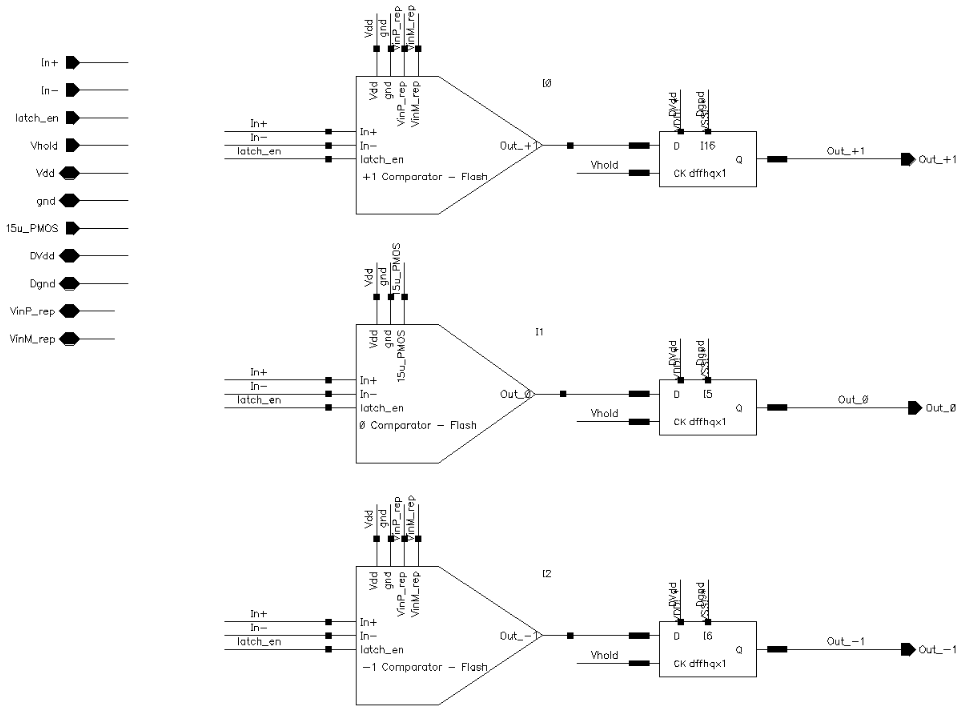


(c) 0 Comparator

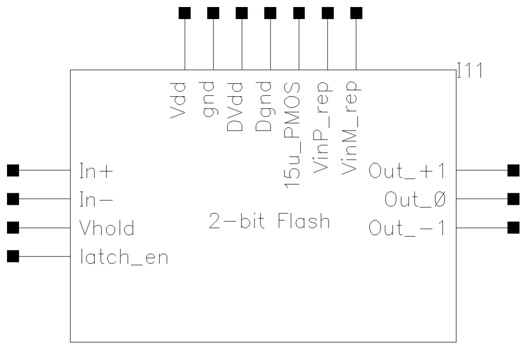
Figure 7.5: 2-bit Flash ADC Comparator Symbols

7.2 Complete 2-bit Flash

The comparators and D-flip-flops were combined to create the 2-bit flash. The schematic and symbol are shown in Figure 7.6.



(a) 2-bit Flash Schematic



(b) 2-bit Flash Symbol

Figure 7.6: 2-bit Flash

Chapter 8

Bias Current Source

8.1 Introduction

The residue amplifier for each stage needs different supply currents. A current bias source was designed to supply these currents to the residue amplifiers on the chip using only one off-chip current source.

8.2 Design

The bias current source was implemented with cascoded PMOS and NMOS transistors. The design consists of the main source and several cascoded MOSFETs that mirror the main source to achieve the desired currents. Figure 8.1presents the complete design of the bias current source.

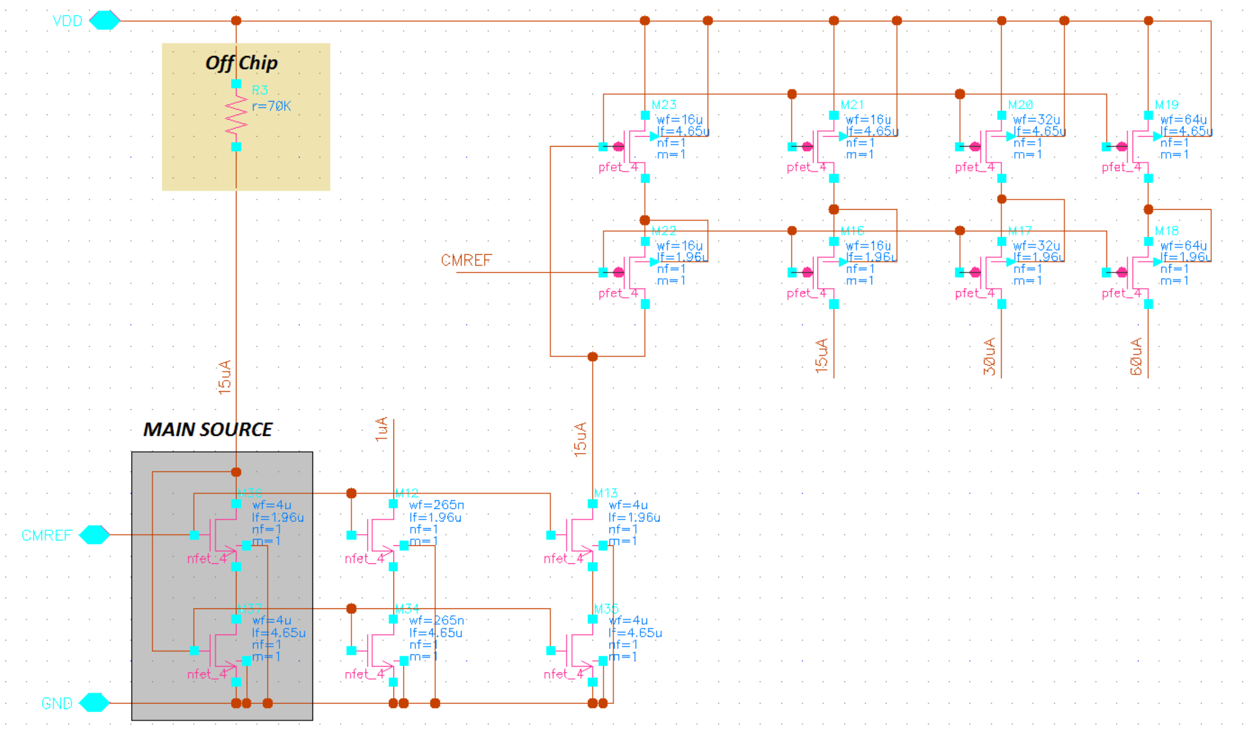


Figure 8.1: Bias Current Source

As shown in the Figure 8.1, the resistor is not part of the actual chip design of the bias current source. This resistor is off the chip and is connected to the power supply, $V_{DD} = 1.8V$ to generate the required current. The other side of the resistor is wired to the main source in the chip. The resistor value was chosen to be $70k\Omega$ in order to obtain 15μA flowing through the main source. The rest of the cascoded mirrors were implemented to achieve the desired supply current for the residue amplifier at different stages. The Table 8.1 shows the corresponding W/Ls and currents that the bias current source provides.

Bias Current For:	Mirror	Width (um)	Length(um)	Current (uA)
Main source	NMOS(top)	4	1.96	15
	NMOS(bottom)	4	4.65	
Smallest Stage with C_L = 125fF	PMOS(top)	16	4.65	15
	PMOS(bottom)	16	1.96	
Stage with C_L = 250fF	PMOS(top)	32	4.65	30
	PMOS(bottom)	32	1.96	
Largest Stage with C_L = 500fF	PMOS(top)	64	4.65	60
	PMOS(bottom)	64	1.96	
Current Mirror (from Residue Amplifier Schematic)	NMOS(top)	0.265	1.96	1
	NMOS(bottom)	0.265	4.65	

Table 8.1: Bias Current Source Currents

8.3 Simulation

In order to verify the bias current source is supplying all the correct currents, it was tested with the residue amplifier. As a result, it was observed that the residue amplifier supplied with the bias current source provides the correct gain, output common mode voltage, and bandwidth.

8.4 Layout

The layout for the bias circuit was designed by dividing each device in half. The layout for half of the bias was completed and then connected in parallel with a mirrored second half. This division and symmetry reduces mismatch without requiring the complexity of a centroided layout.

Chapter 9

Timing Design

9.1 Introduction

The timing controls all of the switch in the switches capacitor circuit as well as the comparator. In future implementations of this design, the timing with be implemented on the FPGA that will control the digital calibration. In order to design the timing for this circuit without the FPGA, both external sources and internal combinational logic were used. There are eighteen timing signals needed, but the number of external sources is limited to three because of a limited number of input pins. These eighteen timing signals can be seen below in Figure 9.1.

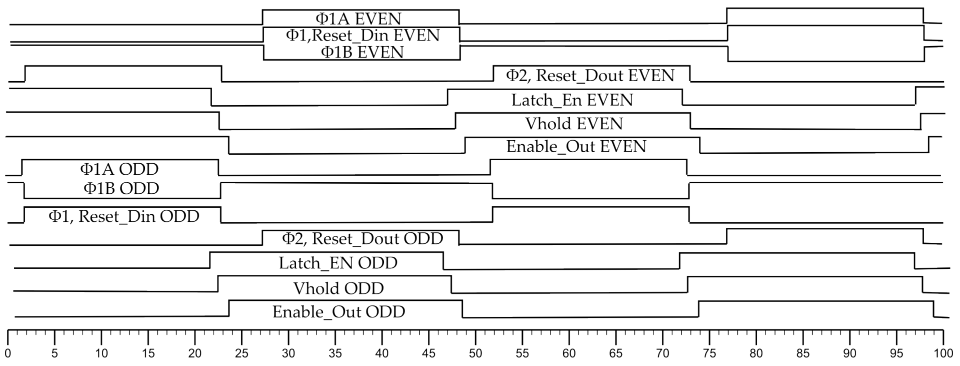


Figure 9.1: Timing Diagram

9.2 Design

The signals that were chosen to be externally feed into the ADC are Φ1 ODD, Φ1 EVEN, and Latch_Enable ODD. The design of the circuit to create the timing signals was simplified by the two phases and even and odd stages. Many of the signals are the same, such as Φ2 ODD, Reset_Dout ODD, and Φ1 EVEN.

In Figure 9.2, the outputs on the right of the circuit are all the timing of the switch capacitor circuit. These signals are all some variation of Φ1 odd and even. The design of Φ1A is important to the correct operation of the switched capacitor array. This signal is important in eliminating charge injection and it needs to start before and end before Φ1. This is done by using a NAND gate connected to the input and a delayed version of the input. The delayed input becomes the advanced signal and the output of the NAND gate is now Φ1B and can be inverted to get Φ1. The simulation of the switch capacitor timing is shown below in Figure 9.3. The 144x output buffer was chosen based on the size of the MOSFET that each signal is driving. This buffer size offered the best rise time for both NMOS and PMOS switches.

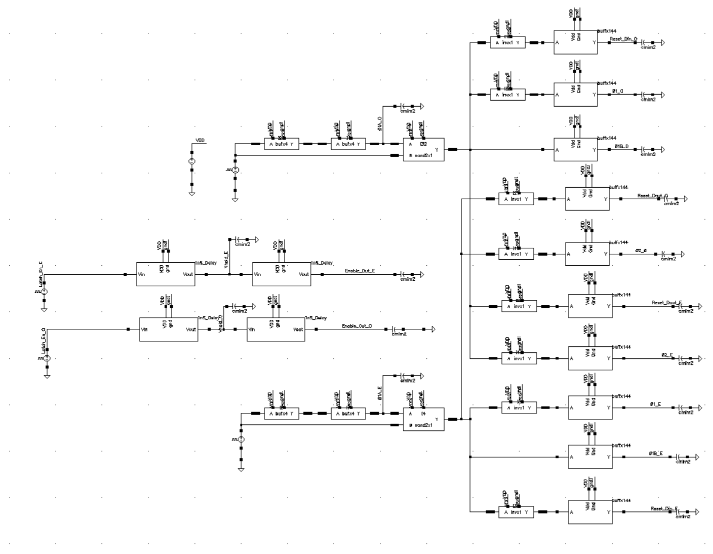


Figure 9.2: Timing Schematic

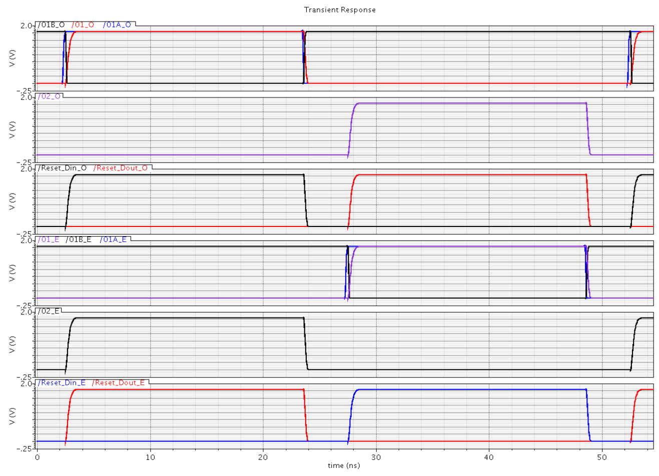


Figure 9.3: Switch Capacitor Timing

The circuit shown the center of Figure 9.2generates the comparator timing. Because the comparator requires three signals that are separated by about one nanosecond, this is done with buffers that were sized to create the needed delay. The schematic shows separate signals driving each but because of the $25nS$ pulse width, the signal can be inverted to generate the even stage. Figure 9.4shows the generated comparator timing signal and also includes Φ1, and Φ2 for a reference.

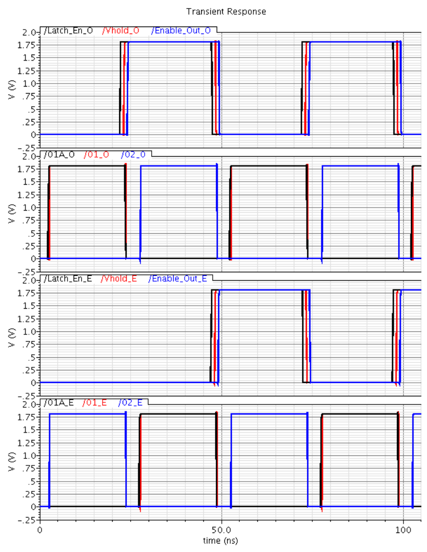
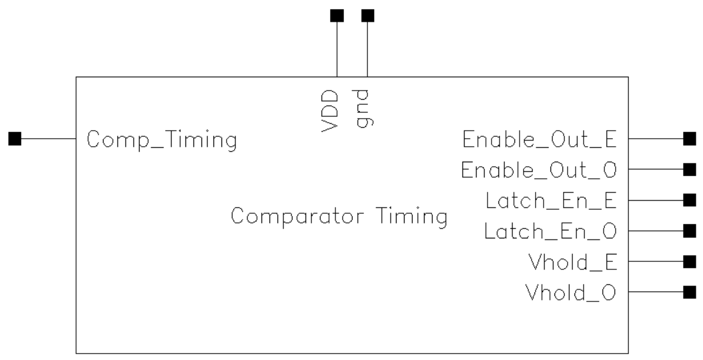
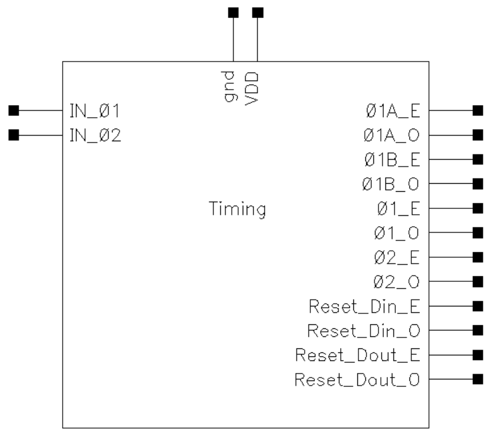


Figure 9.4: Comparator Timing

The timing circuit was combined into two separate timing blocks, switch capacitor timing and comparator timing. These are separate due to scaling only being applied to the switch capacitor and residue amplifier and not the comparator.



(a) Comparator



(b) Switch Capacitor

Figure 9.5: Timing Blocks

Chapter 10

Results and Conclusions

The blocks from each sub-system, described in the design sections, were combined in a full system to characterize the ADC performance. In order to measure the ADCs performance, the original design goals and specifications were compared to the results of the simulated design. Signal-to-noise ratio and figure of merit were also examined. Also, the results of the correction algorithm is simulated using simulation data in MATLAB.

10.1 Full Circuit Simulations

A schematic representation of each block connected as a full stage is shown below in Figure 10.1.

A pipeline ADC can be evaluated based on its stage to stage transfer function, or residue plot. The ideal residue plot for the 1.5 Bit ADC can be seen in the Background section, 2, in Figure 4.8. In order to ensure correct operation for all of the stages, the residue plot from each of the ten simulated stages was included in Figure 10.2. In this figure, each stage is represented by a different color. It can be seen that there is no stage to stage variation. The simulation has data from 195 conversions over $10us$, using an input waveform that is sweep from $-0.6V$ to $+0.6V$. The residue plot also shows the non-linearity from the residue amplifier towards the limits of the input range.

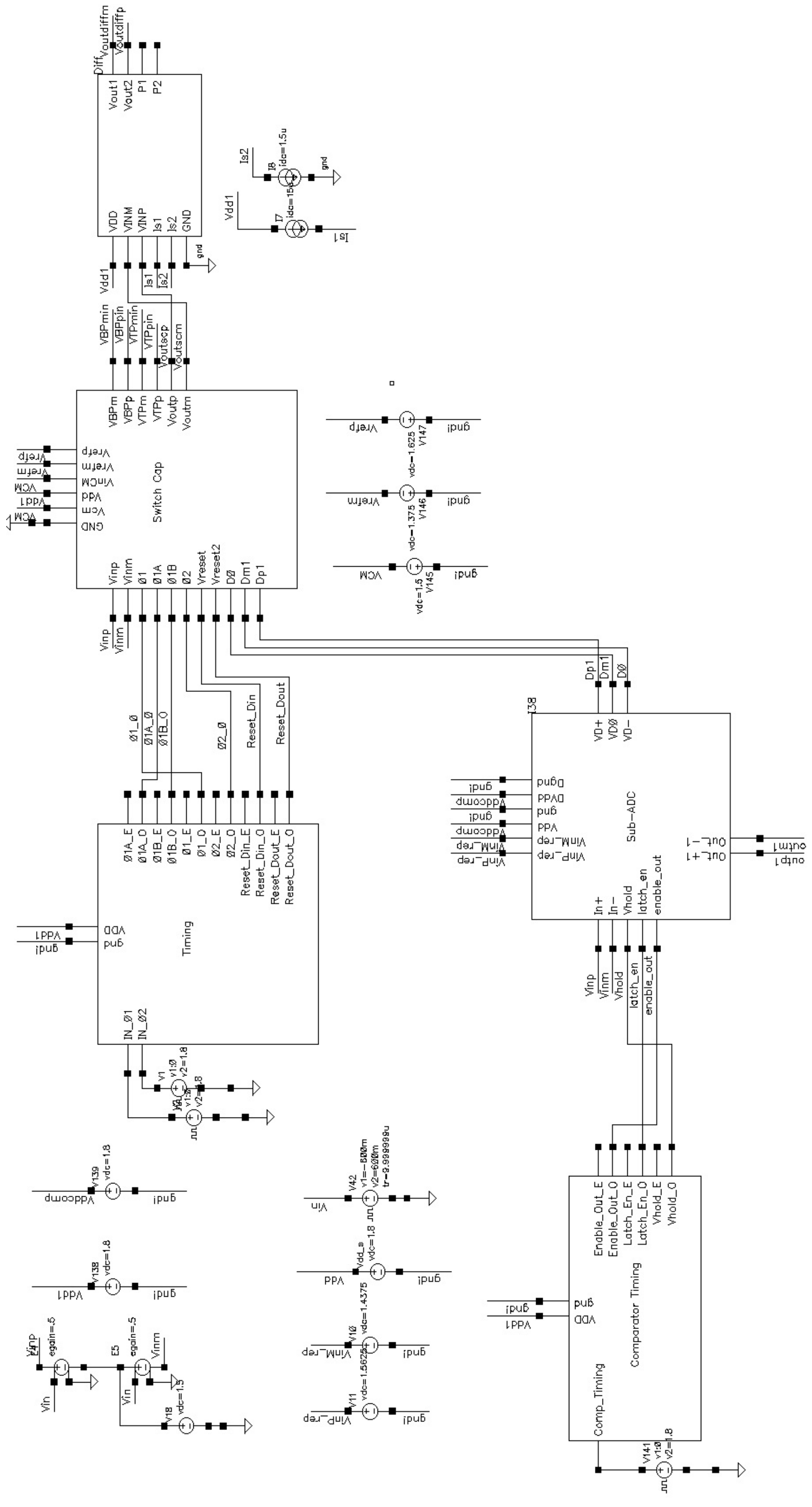


Figure 10.1: Full Stage Schematic

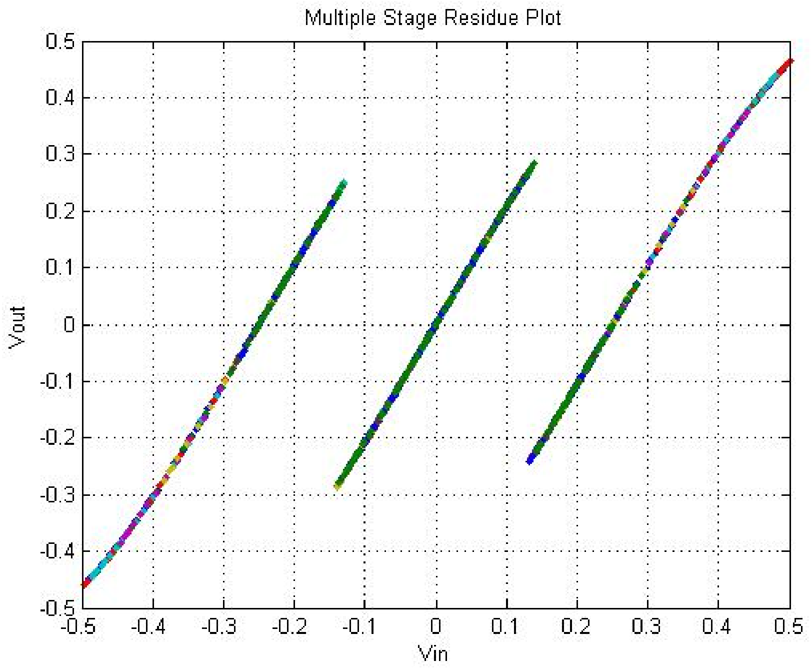


Figure 10.2: Full System Residue

The waveforms used to generate this residue plots experienced the effects of charge injection. This charge injection can be seen in 10.3. This charge injection increases around the decision levels $\pm V_{REF}/4$, or $\pm 0.125V$. This means that the charge injection is dependent on the decision and the voltage across the sampling capacitor. The effects of this charge injection are not noticeable in the residue plot.

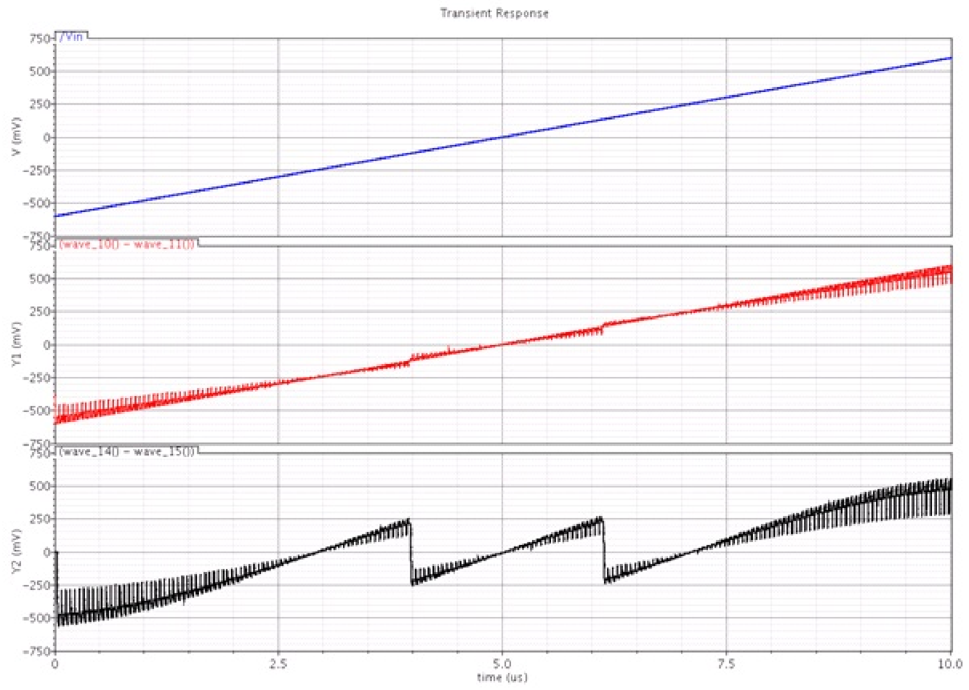


Figure 10.3: Stage One Residue Plot Input and Output

10.2 Power

One goal of this work was to design an ADC with low power consumption. Several techniques were employed to help reduce the power requirement including not using a separate input sample and hold circuit, non-linear residue stage and scaling the size of the stages. The scaled portion of the design, the switch capacitor array and differential pair, together drew about $1mA$ in the smallest stage. The comparator, which was not scaled, drew $1.36mA$. If we multiply our scaled current by $8+4+2+1+1+1+1+1+1+1 = 21$ and the comparator current by 10 stages, and that total by the supply voltage of $1.8V$ the power of $62mW$ is found. This value is an estimate for the total power because it does not include reference voltage currents and it multiplies the differential pair current by more than is should due to not having identical scaling to the switch capacitor circuit.

$$P = IV$$

$$P_{total} = (21 \cdot 1mA + 10 \cdot 1.36mA)1.8V = 62mW \tag{10.1}$$

This power is higher than the goal power of 1mW. Future work will help reduce this power consumption.

10.3 Correction

In order to correct the non-linearities caused by the open loop residue amplifier, a correction algorithm was implemented using a look-up table (LUT). In this work, the algorithm was implemented in MATLAB using the simulated results. This correction will, in future work, be implemented on an FPGA in digital hardware. The correction algorithm uses the comparator decisions and the LUT created based on the residue characteristic to determine the original input. The data from the full system residue plot in Figure 10.2 was interpolated to create three LUTs with 2^6 data points, each for each comparator decision. For the digital hardware implementation, a smaller amount of points may need to be used to conserve memory in the FPGA. The inverse LUT is shown in Figure 10.4.

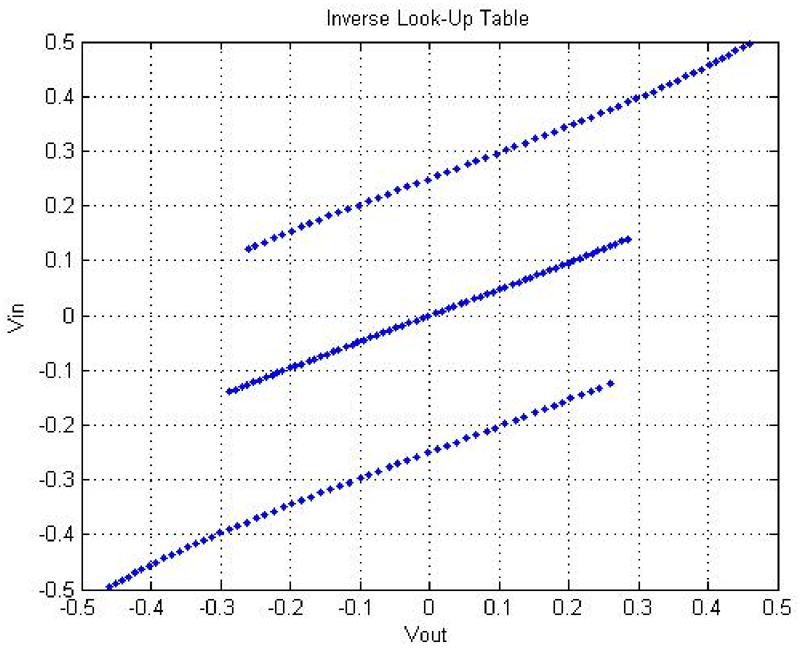


Figure 10.4: Inverse Look-Up Table

If a stage of the pipeline ADC is modeled as a function, [19]

$$V_{RESIDUE} = f(\frac{V_{IN}}{V_{REF}}, D) \quad (10.2)$$

The output of the ADC then becomes,

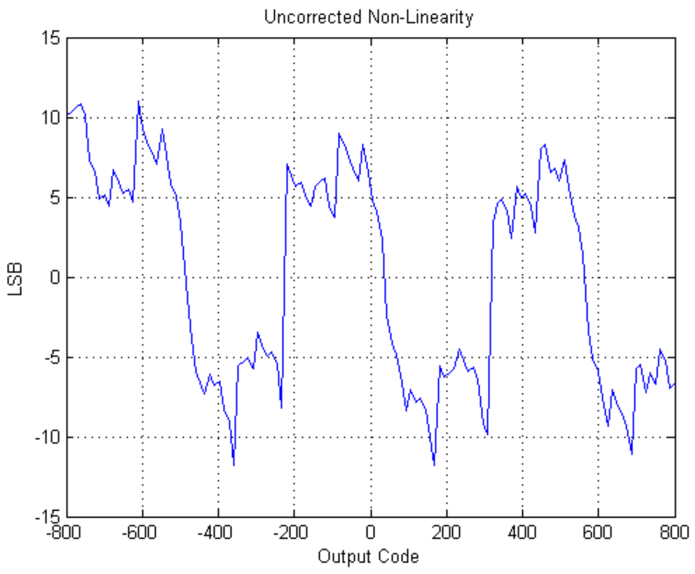
$$x = f(f(f(V_{IN}, D_1), D_2), D_3) \quad (10.3)$$

In correction, the inverse function is used to compute the input,

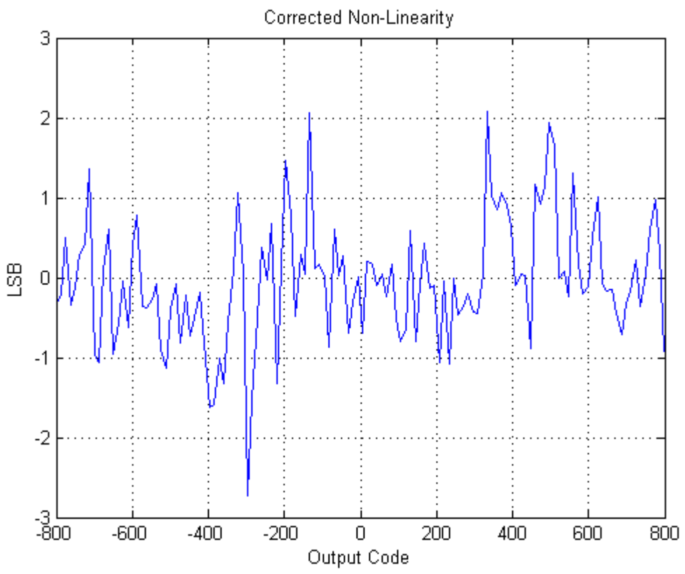
$$x = f^{-1}(D_1, f^{-1}(D_2, f^{-1}(D_3, V_{RESIDUE}))) \quad (10.4)$$

The final result from the simulation and implementation of the correction algorithm is shown in Figure 10.6. The dashed, green waveform of the uncorrected data shows two main non-linearities that are caused by the residue amplifier and by the charge injection. Once the correction is done using the inverse LUT, the output code is corrected and the non-linearities were eliminated. To quantify the improvement in non-linearity, a plot of the LSB difference from a linear line with the same slope was done for both the corrected and uncorrected output, see Figure 10.5. The uncorrected non-linearity is $\pm 10 LSBs$ and the correction improves this by a factor of five to $\pm 2LSBs$. When first analyzing the data, it was found that there were decisions out of the range on the LUT. This is due to imperfect centering at the decision levels $\pm 0.125V$and not enough overlapping of the decisions. This was corrected by extrapolating further into the zero decision, both from the plus and minus decision. This extrapolation may be to blame for the large peaks of the corrected non-linearity near the decisions levels.

When comparing the corrected and the uncorrected output codes, the nonlinearities from the residue amplifier are clearly corrected. Both the corrected and uncorrected output codes show deviation from the original input form. At the lower values, the output code is perfectly aligned with the input waveform; however, as the output voltage increases, the deviation also increases. The simulation that generated these results ramped the input up over time. The results seem to imply that as time progresses, the output code gets further and further from the actual input. This means that there is something slowly decreasing the overall output value. This may be caused by a problem with the reset switching that causes values from previous conversions to carry into the next conversion. This could also possibly be due to gain



(a) Uncorrected



(b) Corrected

Figure 10.5: Non-linearity Comparison

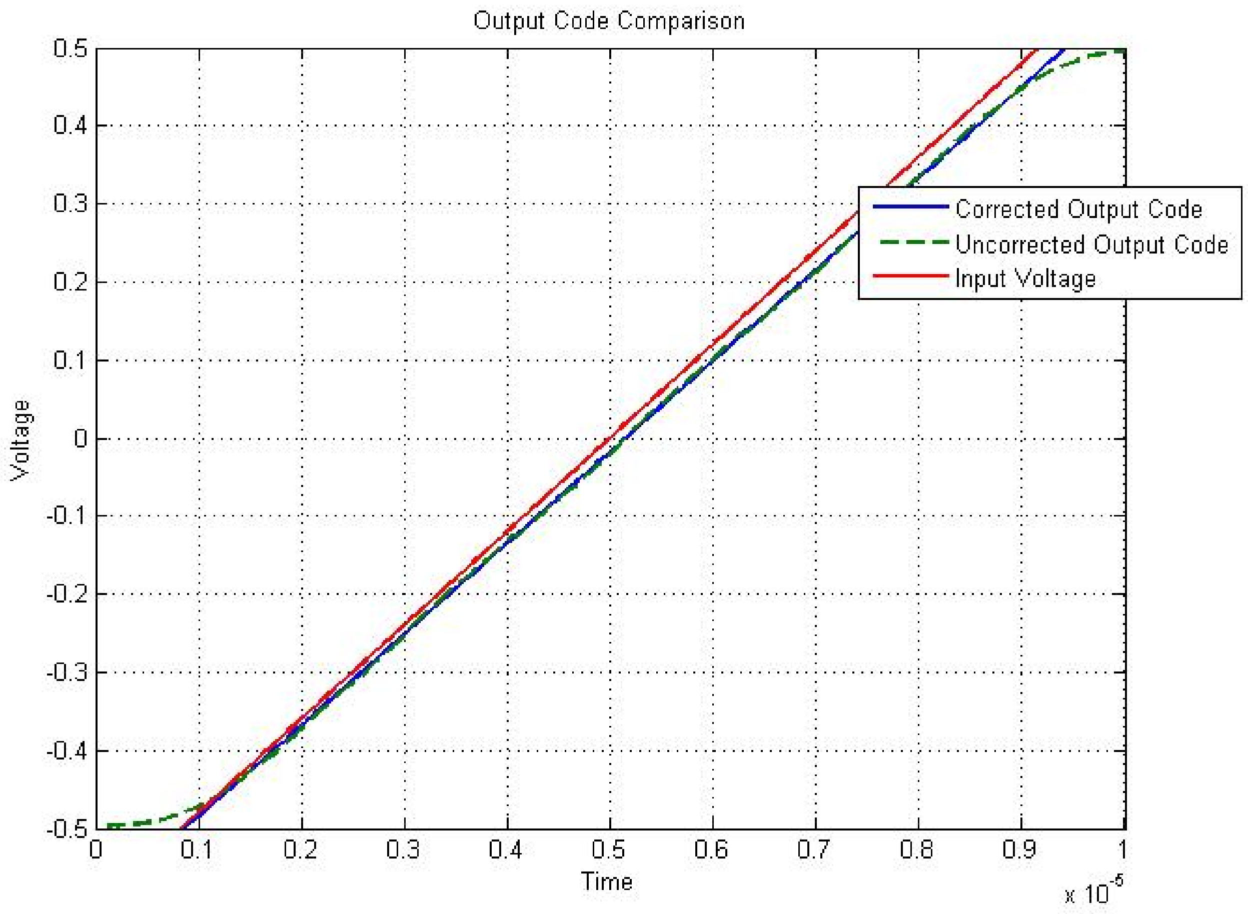


Figure 10.6: Compared Output Code

10.4 Signal-to-Noise Ratio and Figure of Merit

The signal-to-noise ratio (SNR) is the ratio of the input voltage to the noise. The design decision to reduce the differential signal swing from 1V to 0.5V reduces the SNR of the ADC. As discussed in the design sections for the switched capacitor network, 4.2, and the residue amplifier, 5.4, the noise of the circuit for the largest stage is $90\mu V$ for the switched capacitor circuit and $245\mu V$ for the residue amplifier. The total noise was $335\mu V$. The SNR was calculated as follows:

$$SNR = \frac{\mu}{\sigma} = 20log(\frac{352mV}{335uV}) = 60.4dB \tag{10.5}$$

$$SNR_{IDEAL} = 6.02N + 1.76dB = 74dB \tag{10.6}$$

The resulting SNR, 60.4dB, is lower than the ideal value of 74dB for a 12-Bit ADC. This lower SNR decreases the effective number of bits (ENOB) and affects the Figure of Merit (FOM). The computations for the ENOB and FOM are shown below.

$$ENOB = \frac{SNR - 1.76}{6.02} = 9.74Bits \tag{10.7}$$

$$FOM = \frac{Power}{(2^N)(f_s)} = \frac{62mW}{2^{9.74} \times 20MHz} = 3.63pJ/step \tag{10.8}$$

The Figure of Merit chart in Figure 10.7 shows the design in this work compared to other pipeline ADCs. Despite not meeting some design goals, this design is competitive with similar ADCs. Recommendations on how to improve these specifications will be discussed in the Future Work section.

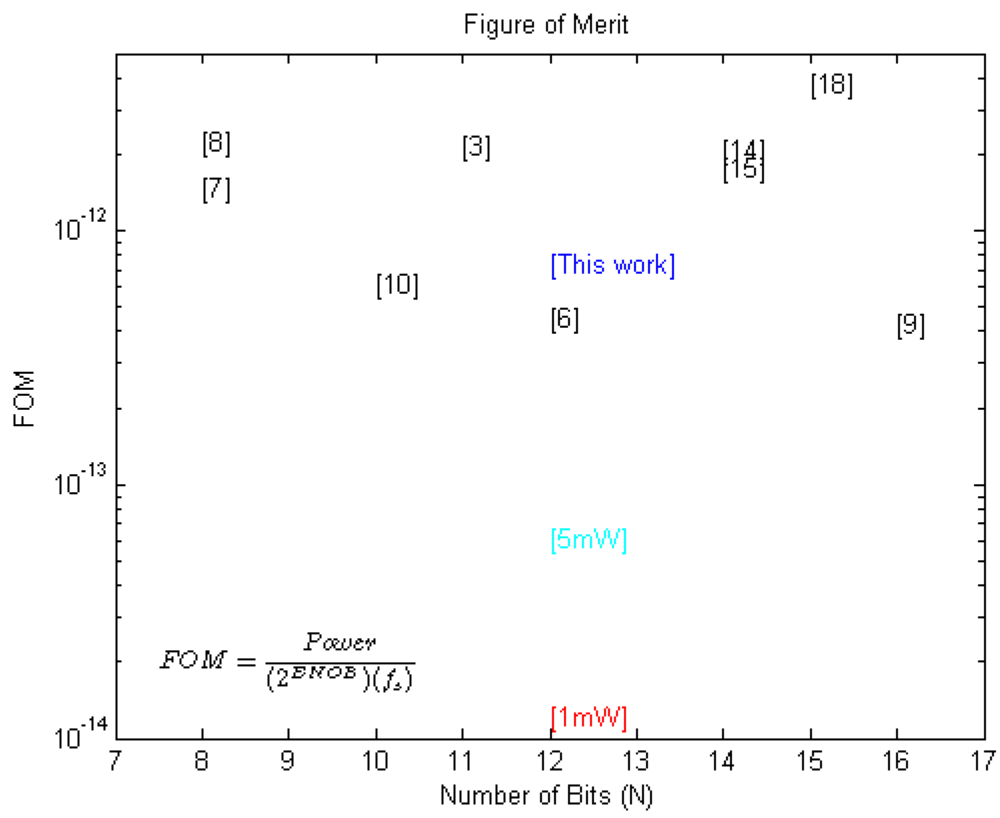


Figure 10.7: Figure of Merit

Chapter 11

Future Work

In this work, the schematic design and simulation of the sub-blocks and full circuit of a 12-bit pipeline ADC were completed. Layout and extracted simulations for several of the sub-blocks were also completed, however, several blocks still remain for which layout needs to be completed. Future work would complete the layout of these remaining blocks of the design. The overall floorplan, tape-out of the design and extracted simulations would also need to be completed before fabrication of the ADC. The completed chip will then need to be tested and characterized with the use of a printed circuit board that will need to be designed and fabricated. Some revisions will also need to be made to the design before fabrication. It is recommended that in the process of making these changes and in future ADC designs that behavioral simulations in SPICE or MATLAB be implemented early in the design process to help identify potential problems earlier in the design process.

11.1 Reduction of Power Consumption

The power consumption of the design described in this report is 62mW, much higher than the target power of 1mW. Because the comparator is the sub-block that consumes the most power currently, a redesign of portions of this circuit could help reduce power consumption. The preamplifier could be redesigned to require less current. Lowering the current used in the preamplifier would require the mismatch in the devices to be larger.

11.2 Improving SNR and ENOB

The signal-to-noise ratio of the design could be reduced in future work. The SNR of the ADC depends primarily on the noise from the switched capacitor array and the residue amplifier. To reduce the noise, the main differential pair in the residue amplifier could be modified so that the devices in the amplifier are larger, as discussed in the design section of the residue amplifier. The size of the capacitors in the switched capacitor network could also be increased to reduce noise; the capacitors would need to be sized for both the desired noise and the desired speed.

11.3 Continuing Correction and Calibration Implementation

This work includes an implementation of the correction algorithm in MATLAB. The results of this implementation show a discrepancy between the original input values and the corrected values, described in the Results section of this report. The cause of this difference will need to be investigated in future work. One possible cause could be a problem in the reset switching and charge injection that results in information being carried between conversions. In future work this calibration would also be implemented in digital hardware, such as on an FPGA.

This work includes the design of a single ADC as the split-ADC calibration technique was not fully implemented and tested. Future work towards implementing a pipeline ADC with a split-ADC architecture would include the fabrication of the single ADC described in this work and test the calibration with a simulated ideal second ADC. Eventually, a second ADC with a different residue characteristic would be designed for the complete implementation of the pipeline ADC with a split-ADC architecture.

Appendix A

MATLAB Correction Code

```
%% Correction

%% Get data from csv file
data = csvread('ALLFINAL.csv' );

%% Place different columns of the csv files into seperate arrays
time = data(:,1);
Vin = data(:,5);
Vdd_current = data(:,2);
Vdd_compcurrent = data(:,4);
Vdd_stage1current = data(:,3);

%% data 6-28 are digital outputs even being the plus and odd are minus
outp1 = data(:,6);
outm1 = data(:,7);
outp2 = data(:,8);
outm2 = data(:,9);
outp3 = data(:,10);
outm3 = data(:,11);
outp4 = data(:,12);
outm4 = data(:,13);
outp5 = data(:,14);
outm5 = data(:,15);
outp6 = data(:,16);
outm6 = data(:,17);
outp7 = data(:,18);
outm7 = data(:,19);
outp8 = data(:,20);
outm8 = data(:,21);
outp9 = data(:,22);
outm9 = data(:,23);
outp10 = data(:,24);
outm10 = data(:,25);
out2bitp1 = data(:,26);
out2bit0 = data(:,27);
out2bitm1 = data(:,28);

%% sort output values(one value per decision)
%% every 25nS. start each stage at 25 points later.
```

```
Outp1 = outp1(25:50:10000);
Outm1 = outm1(25:50:10000);
Out1 = zeros(200,1);

%% use function called digital to change close to zero=zero
%% and close to 1.8 = 1
%% Then look at outp and outm and transfer to single -1,0,or +1
for j=1:195
    Outp1(j) = digital(Outp1(j));
    Outm1(j) = digital(Outm1(j));
    if ((Outp1(j)==0)&&(Outm1(j)==0))
        Out1(j) = 0; % 01
    elseif ((Outp1(j)==0) && (Outm1(j)==1))
        Out1(j) = -1; % 00
    elseif ((Outp1(j)==1) && (Outm1(j)==0))
        Out1(j) = 1; % 10
    else
        Out1(j) = 3; % error
    end

end
%% repeat for all stages.
Outp2 = outp2(50:50:10000);
Outm2 = outm2(50:50:10000);
Out2 = zeros(200,1);
for j=1:195
    Outp2(j) = digital(Outp2(j));
    Outm2(j) = digital(Outm2(j));
    if ((Outp2(j)==0)&&(Outm2(j)==0))
        Out2(j) = 0; % 01
    elseif ((Outp2(j)==0) && (Outm2(j)==1))
        Out2(j) = -1; % 00
    elseif ((Outp2(j)==1) && (Outm2(j)==0))
        Out2(j) = 1; % 10
    else
        Out2(j) = 3; % error
    end

end

Outp3 = outp3(75:50:10000);
Outm3 = outm3(75:50:10000);
Out3 = zeros(200,1);
for j=1:195
    Outp3(j) = digital(Outp3(j));
    Outm3(j) = digital(Outm3(j));
    if ((Outp3(j)==0)&&(Outm3(j)==0))
        Out3(j) = 0; % 01
    elseif ((Outp3(j)==0) && (Outm3(j)==1))
        Out3(j) = -1; % 00
    elseif ((Outp3(j)==1) && (Outm3(j)==0))
        Out3(j) = 1; % 10
    else
        Out3(j) = 3; % error
    end

end

Outp4 = outp4(100:50:10000);
Outm4 = outm4(100:50:10000);
Out4 = zeros(200,1);
for j=1:195
    Outp4(j) = digital(Outp4(j));
    Outm4(j) = digital(Outm4(j));
    if ((Outp4(j)==0)&&(Outm4(j)==0))
        Out4(j) = 0; % 01
    elseif ((Outp4(j)==0) && (Outm4(j)==1))
        Out4(j) = -1; % 00
    elseif ((Outp4(j)==1) && (Outm4(j)==0))
        Out4(j) = 1; % 10
    else
        Out4(j) = 3; % error
    end

end

Outp5 = outp5(125:50:10000);
Outm5 = outm5(125:50:10000);
Out5 = zeros(200,1);
for j=1:195
    Outp5(j) = digital(Outp5(j));
    Outm5(j) = digital(Outm5(j));
    if ((Outp5(j)==0)&&(Outm5(j)==0))
        Out5(j) = 0; % 01
    elseif ((Outp5(j)==0) && (Outm5(j)==1))
        Out5(j) = -1; % 00
    elseif ((Outp5(j)==1) && (Outm5(j)==0))
        Out5(j) = 1; % 10
    else
        Out5(j) = 3; % error
    end

end

Outp6 = outp6(150:50:10000);
Outm6 = outm6(150:50:10000);
Out6 = zeros(200,1);
for j=1:195
    Outp6(j) = digital(Outp6(j));
    Outm6(j) = digital(Outm6(j));
    if ((Outp6(j)==0)&&(Outm6(j)==0))
        Out6(j) = 0; % 01
    elseif ((Outp6(j)==0) && (Outm6(j)==1))
        Out6(j) = -1; % 00
    elseif ((Outp6(j)==1) && (Outm6(j)==0))
        Out6(j) = 1; % 10
    else
        Out6(j) = 3; % error
    end

end

Outp7 = outp7(175:50:10000);
Outm7 = outm7(175:50:10000);
Out7 = zeros(200,1);
for j=1:195
    Outp7(j) = digital(Outp7(j));
    Outm7(j) = digital(Outm7(j));
    if ((Outp7(j)==0)&&(Outm7(j)==0))
        Out7(j) = 0; % 01
    elseif ((Outp7(j)==0) && (Outm7(j)==1))
        Out7(j) = -1; % 00
    elseif ((Outp7(j)==1) && (Outm7(j)==0))
        Out7(j) = 1; % 10
    else
        Out7(j) = 3; % error
    end

end

Outp8 = outp8(200:50:10000);
Outm8 = outm8(200:50:10000);
Out8 = zeros(200,1);
for j=1:195
    Outp8(j) = digital(Outp8(j));
    Outm8(j) = digital(Outm8(j));
    if ((Outp8(j)==0)&&(Outm8(j)==0))
        Out8(j) = 0; % 01
    elseif ((Outp8(j)==0) && (Outm8(j)==1))
        Out8(j) = -1; % 00
    elseif ((Outp8(j)==1) && (Outm8(j)==0))
        Out8(j) = 1; % 10
    else
        Out8(j) = 3; % error
    end

end

Outp9 = outp9(225:50:10000);
Outm9 = outm9(225:50:10000);
Out9 = zeros(200,1);
for j=1:195
    Outp9(j) = digital(Outp9(j));
    Outm9(j) = digital(Outm9(j));
    if ((Outp9(j)==0)&&(Outm9(j)==0))
        Out9(j) = 0; % 01
    elseif ((Outp9(j)==0) && (Outm9(j)==1))
        Out9(j) = -1; % 00
    elseif ((Outp9(j)==1) && (Outm9(j)==0))
        Out9(j) = 1; % 10
    else
        Out9(j) = 3; % error
    end

end

Outp10 = outp10(250:50:10000);
Outm10 = outm10(250:50:10000);
Out10 = zeros(200,1);
for j=1:195
    Outp10(j) = digital(Outp10(j));
    Outm10(j) = digital(Outm10(j));
    if ((Outp10(j)==0)&&(Outm10(j)==0))
        Out10(j) = 0; % 01
    elseif ((Outp10(j)==0) && (Outm10(j)==1))
        Out10(j) = -1; % 00
    elseif ((Outp10(j)==1) && (Outm10(j)==0))
        Out10(j) = 1; % 10
    else
        Out10(j) = 3; % error
    end

end

Out2bitp1 = out2bitp1(275:50:10000);
Out2bit0 = out2bit0(275:50:10000);
Out2bitm1 = out2bitm1(275:50:10000);

%%calculate total uncorrected output
%%multiply descison by the binary value of that stage
%% divide that total by 2^10-1
Total_Out=zeros(195,1);
Voltage_input = zeros(195,1);
for k=1:195
    Total_Out(k)= (2^9)*Out1(k) + (2^8)*Out2(k) + (2^7)*Out3(k) + (2^6)*
        Out4(k) + (2^5)*Out5(k) + (2^4)*Out6(k) + (2^3)*Out7(k) + (2^2)*
        Out8(k) + (2^1)*Out9(k) + Out10(k);
    Voltage_input(k) = ((Total_Out(k)/(2^10-1))*.5); %%take output total
        and divide it by the max output, then multiply by 1 to get input
        value, but since -.5 to .5, subtract .5V
end

%% data 32-41 are each stages capacitor value data
stage1 = data(:,32);
stage2 = data(:,33);
stage3 = data(:,34);
stage4 = data(:,35);
stage5 = data(:,36);
stage6 = data(:,37);
stage7 = data(:,38);
stage8 = data(:,39);
stage9 = data(:,40);
stage10 = data(:,41);

%% start data at correct spot for each stage, parse data
Stage1 = stage1(24:50:9950);
```

```
Stage2 = stage2(49:50:9950);
Stage3 = stage3(74:50:10000);
Stage4 = stage4(99:50:10000);
Stage45 = stage4(99:50:9950);
Stage5 = stage5(124:50:10000);
Stage6 = stage6(149:50:10000);
Stage67 = stage6(149:50:9950);
Stage7 = stage7(174:50:10000);
Stage8 = stage8(199:50:10000);
Stage89= stage8(199:50:9950);
Stage9 = stage9(224:50:10000);
Stage10 = stage10(249:50:10000);

%%make look up table of full residue plot
%%First make one for each decision
%% create xvalues for the LUT
LUT_InM = (-0.55:.425/(2^6):-0.125)';
LUT_In0 = (-.1401:.2792/(2^6):0.1391)';
LUT_InP = (0.121:.429/(2^6):0.55)';
LUT_IN = vertcat( LUT_InM, LUT_In0, LUT_InP);
%% taking just the first stage doesnt give enough data at the decisions to
%% make a clear look up table. So concatenate stages...
STAGES_IN = vertcat(Stage1,Stage2,Stage3,Stage45,Stage5,Stage67,Stage7,
    Stage89,Stage9);
STAGES_OUT = vertcat(Stage2,Stage3,Stage4,Stage5,Stage6,Stage7,Stage8,
    Stage9,Stage10);
[STAGES_INU,index,bb] = unique(STAGES_IN);
STAGES_OUTU = STAGES_OUT(index);
STAGES_ALL = [STAGES_INU, STAGES_OUTU];

%%painfully seperating the different decisions due to overlapp
Minus1_IN = STAGES_INU(1:471);
Minus1_IN([469,467,466,461,458,455,454,449,448])=[];
Minus1_IN(463)= -.125;
Minus1_OUT = STAGES_OUTU(1:471);
Minus1_OUT([469,467,466,461,458,455,454,449,448])=[];
Minus1_OUT(463) = .26;
Zero_IN = STAGES_INU(448:1210);
Zero_IN
    ([3,4,5,6,9,10,12,13,15,16,17,18,21,23,24,747,748,752,755,756,758,759,761,762])
    =[];
Zero_OUT = STAGES_OUTU(448:1210);
Zero_OUT
    ([3,4,5,6,9,10,12,13,15,16,17,18,21,23,24,747,748,752,755,756,758,759,761,762])
    =[];
Plus1_IN = STAGES_INU(1194: 1663);
Plus1_IN([3,4,5,7,8,11,14,17])=[];
Plus1_IN(1)=.121;
Plus1_OUT = STAGES_OUTU(1194: 1663);
Plus1_OUT([3,4,5,7,8,11,14,17])=[];
Plus1_OUT(1)=-.26;

%% inpterperate different decisions LUT over specified range
LUT_Minus1 = interp1(Minus1_IN, Minus1_OUT, LUT_InM, 'linear');
```

```
LUT_Zero = interp1(Zero_IN , Zero_OUT , LUT_In0 , 'linear');
LUT_Plus1 = interp1(Plus1_IN , Plus1_OUT , LUT_InP , 'linear');
LUT_ALL = vertcat(LUT_Minus1 , LUT_Zero , LUT_Plus1);

%%create blank arrays to fill with residue value for correction
Residue10 = zeros(195,1);
Residue9 = zeros(195,1);
Residue8 = zeros(195,1);
Residue7 = zeros(195,1);
Residue6 = zeros(195,1);
Residue5 = zeros(195,1);
Residue4 = zeros(195,1);
Residue3 = zeros(195,1);
Residue2 = zeros(195,1);
Residue1 = zeros(195,1);
Residue0 = zeros(195,1);
Residue10(195)=0;

%%determine which decisions were made ( OUT values aboves)
%%pick the residue
%%use the last stage with 2-bit flash or guess 0
%%guess residue=0 -> that plus the decsion -> residue of prior stage
%%use residue -> repeat
for Conv=195:-1:1
    if (Out10(Conv)==-1)
         Residue9(Conv) = interp1( LUT_Minus1,LUT_InM, Residue10(Conv), '
            linear');
    elseif (Out10(Conv)==0)
        Residue9(Conv) = interp1( LUT_Zero, LUT_In0, Residue10(Conv), '
           linear');
    elseif(Out10(Conv) == 1)
        Residue9(Conv) = interp1( LUT_Plus1, LUT_InP,Residue10(Conv), '
           linear');
    end

    if (Out9(Conv)==-1)
         Residue8(Conv) = interp1( LUT_Minus1, LUT_InM, Residue9(Conv), '
            linear');
    elseif (Out9(Conv)==0)
        Residue8(Conv) = interp1( LUT_Zero, LUT_In0, Residue9(Conv), '
           linear');
    elseif(Out9(Conv) == 1)
        Residue8(Conv) = interp1(LUT_Plus1, LUT_InP, Residue9(Conv), '
           linear');
    end

    if (Out8(Conv)==-1)
         Residue7(Conv) = interp1(LUT_Minus1,LUT_InM, Residue8(Conv), '
            linear');
    elseif (Out8(Conv)==0)
        Residue7(Conv) = interp1( LUT_Zero,LUT_In0, Residue8(Conv), '
           linear');
    elseif(Out8(Conv) == 1)
        Residue7(Conv) = interp1( LUT_Plus1, LUT_InP,Residue8(Conv), '
            linear');
    end

    if (Out7(Conv)==-1)
         Residue6(Conv) = interp1( LUT_Minus1,LUT_InM, Residue7(Conv), 'linear');
    elseif (Out7(Conv)==0)
        Residue6(Conv) = interp1( LUT_Zero,LUT_In0, Residue7(Conv), 'linear');
    elseif(Out7(Conv) == 1)
        Residue6(Conv) = interp1( LUT_Plus1, LUT_InP,Residue7(Conv), 'linear');
    end

    if (Out6(Conv)==-1)
         Residue5(Conv) = interp1( LUT_Minus1, LUT_InM,Residue6(Conv), 'linear');
    elseif (Out6(Conv)==0)
        Residue5(Conv) = interp1( LUT_Zero,LUT_In0, Residue6(Conv), 'linear');
    elseif(Out6(Conv) == 1)
        Residue5(Conv) = interp1( LUT_Plus1,LUT_InP, Residue6(Conv), 'linear');
    end

    if (Out5(Conv)==-1)
         Residue4(Conv) = interp1( LUT_Minus1,LUT_InM, Residue5(Conv), 'linear');
    elseif (Out5(Conv)==0)
        Residue4(Conv) = interp1( LUT_Zero, LUT_In0,Residue5(Conv), 'linear');
    elseif(Out5(Conv) == 1)
        Residue4(Conv) = interp1( LUT_Plus1,LUT_InP, Residue5(Conv), 'linear');
    end

    if (Out4(Conv)==-1)
         Residue3(Conv) = interp1(LUT_Minus1, LUT_InM, Residue4(Conv), 'linear');
    elseif (Out4(Conv)==0)
        Residue3(Conv) = interp1( LUT_Zero, LUT_In0,Residue4(Conv), 'linear');
    elseif(Out4(Conv) == 1)
        Residue3(Conv) = interp1( LUT_Plus1, LUT_InP,Residue4(Conv), 'linear');
    end

    if (Out3(Conv)==-1)
         Residue2(Conv) = interp1( LUT_Minus1, LUT_InM,Residue3(Conv), 'linear');
    elseif (Out3(Conv)==0)
        Residue2(Conv) = interp1(LUT_Zero, LUT_In0, Residue3(Conv), 'linear');
    elseif(Out3(Conv) == 1)
```

```
        Residue2(Conv) = interp1( LUT_Plus1,LUT_InP, Residue3(Conv), '
            linear');
    end

    if (Out2(Conv)==-1)
         Residue1(Conv) = interp1(LUT_Minus1, LUT_InM,Residue2(Conv), '
             linear');
    elseif (Out2(Conv)==0)
        Residue1(Conv) = interp1( LUT_Zero, LUT_In0,Residue2(Conv), '
            linear');
    elseif(Out2(Conv) == 1)
        Residue1(Conv) = interp1( LUT_Plus1, LUT_InP,Residue2(Conv), '
            linear');
    end

    if (Out1(Conv)==-1)
         Residue0(Conv) = interp1( LUT_Minus1,LUT_InM, Residue1(Conv), '
             linear');
    elseif (Out1(Conv)==0)
        Residue0(Conv) = interp1( LUT_Zero,LUT_In0, Residue1(Conv), '
            linear');
    elseif(Out1(Conv) == 1)
        Residue0(Conv) = interp1( LUT_Plus1,LUT_InP, Residue1(Conv), '
            linear');
    end
end


%%variables for plotting
Min = 10;
Max = 180;
tforresults= .05E-6:10E-6/195:10E-6;
TIME = 1E-9:1E-9:10E-6;
VIN = interp1(TIME, Vin, tforresults, 'linear');
Residue0error = Residue0(Min:1:Max);
timefliped = tforresults(:);
timefliped = timefliped(Min:1:Max);
uncorrected = Voltage_input(Min:1:Max);

Code = VIN(Min:1:Max)./(.5/2^10);

%%error
[p,s] = polyfit(timefliped, Residue0error,1);
bestfit = polyval(p,timefliped);
figure(5)
plot(timefliped,bestfit)
errorcorrected = (bestfit-Residue0error)./(.5/2^10);
erroruncorrected = (bestfit-uncorrected)./(.5/2^10);


figure(6)
plot(Code,errorcorrected)
grid on
axis([-800 800 -3 3])
```

```
xlabel('Output␣Code')
ylabel('LSB')
title('Corrected␣Non-Linearity')

figure(7)
plot(Code,erroruncorrected)
grid on
axis([-800 800 -15 15])
xlabel('Output␣Code')
ylabel('LSB')
title('Uncorrected␣Non-Linearity')

%% compared outputs to input
figure(1)
plot(tforresults,Residue0,tforresults, Voltage_input, tforresults, VIN)
grid on
axis([0 1E-5 -.5 .5])
xlabel('Time')
ylabel('Voltage')
title ('Output␣Code␣Comparison')

%%LUT
figure(2)
plot( LUT_IN, LUT_ALL, '.');
grid on
axis([-.5 .5 -.5 .5])
xlabel('Vin')
ylabel('Vout')
title('Look-Up␣Table')

%%inverse LUT
figure(3)
plot(LUT_ALL, LUT_IN, '.')
grid on
axis([-.5 .5 -.5 .5])
xlabel('Vout')
ylabel('Vin')
title('Inverse␣Look-Up␣Table')

%%Full Residue
figure(4)
plot(Stage1, Stage2, '.', Stage2, Stage3, '.',Stage3, Stage4, '.', Stage45
    , Stage5, '.', Stage5, Stage6, '.',Stage67, Stage7, '.',Stage7, Stage8,
     '.',Stage89, Stage9, '.', Stage9, Stage10, '.' )
xlabel('Vin')
ylabel('Vout')
axis([-.5 .5 -.5 .5])
title('Multiple␣Stage␣Residue␣Plot')
grid on
```

Bibliography

[1] Imran Ahmed. *Pipelined ADC Design and Enhancement Technique*. 2010.

[2] Imran Ahmed and David A. Johns. Dac nonlinearity and residue gain error correction in a pipelined adc using a split-adc architecture. *Research in Microelectronics and Electronics 2006, Ph. D.*, 2006.

[3] Paul J. Hurst Carl R. Grace and Stephen H. Lewis. A 12-bit 80-msample/s pipelined adc with bootstrapped digital calibration. *IEEE Journal of Solid-State Circuits*, 40(5):1038–1046, 2005.

[4] Analog Devices. *Analog-Digital Conversion Handbook*. Third edition, 1985.

[5] Charles Gammal Devin Auclair and Fitzgerald Huang. 12b 100msps pipeline adc with open-loop reisdue amplifier, 2008.

[6] Hee-Cheol Choi et al. A 1.2-v 12-b 120-ms/s sha-free dual-channel nyquist adc based on midcode calibration. *IEEE Transactions of Solid-State Circuits*, 56(5):894–911, 2009.

[7] J. Olivieira et al. An 8-bit 120-ms/s interleaved cmos pipeline adc based on mos parametric amplification. *IEEE Transactions on Circuits and Systems*, 57(2):105–109, 2010.

[8] Jongwoo Lee et al. A 2.5 mw 80 db dr 36 db sndr 22 ms/s logarithmic pipeline adc. *IEEE Transactions on Circuits and Systems*, 44(10):2755–2765, 2009.

[9] Siddharth Devarajan et al. A 16-bit, 125 ms/s, 385 mw, 78.7 db snr cmos pipeline adc. *IEEE Journal of Solid-State Circuits*, 21(44):3305–3313, 2009.

[10] Si-Wook Yoo Sun-Young Hwang Seung-Hoon Lee Hee-Cheol Choi, Young-Ju Kim. A programmable 0.8-v 10-bit 60-ms/s 19.2-mw 0.13um cmos adc operating down to 0.5v. *IEEE Transactions on Circuits and Systems*, 55(4):319–323, 2008.

[11] Echere Iroaga and Boris Murmann. A 12-bit 75-ms/s pipelined adc using incomplete settling. *IEEE Journal of Solid-State Circuits*, 42(4), 2007.

[12] Sanjeev Goluguri John A. McNeill and Abhilash Nair. "split-adc" digital background correction of open-loop residue amplifier nonlinearity errors in a 14b pipeline adc. 2007.

[13] David A. Johns and Ken Martin. *Analog Integrated Circuit Design*. 1997.

[14] K.J. Moon Y.L. Kim K.H.Lee, H.C. Choi and S.H. Lee. Calibration-free 14b 70ms/s 0.13um cmos pipeline a-d converters based on high-matching 3d symmetric capacitors. *Electronics Letters*, 43(6), 2007.

[15] K.S. Kim K.H.Lee, Y.J. Kim and S.H. Lee. 14 bit 50 ms/s 0.18 um cmos pipeline adc based on digital error calibration. *Electronics Letters*, 45(21), 2009.

[16] Boris Murmann and Bernhard E. Boser. *Digitally Assisted Pipeline ADCs, Theory and Implementation*. Boston, Massachusetts, 2004.

[17] Shant Orchanian. Split non-linear cyclic analog-to-digital converter. Master's thesis, Worcester Polytechnic Institute, 2010.

[18] Yun-Shiang Shu and Bang-Sup Song. A 15-bit linear 20-ms/s pipelined adc digitally calibrated with signal dependent dithering. *IEEE Journal of Solid-State Circuits*, 43(2):342–350, 2008.

[19] Hattie Spetla. Split cyclic analog to digital converter using a nonlinear gain stage. Master's thesis, Worcester Polytechnic Institute, 2009.

[20] R. Ramachandran-T.R. Viswanathan Stephen H. Lewis, Scott Fetterman. A 10-b 20-msample/s analog-to-digital converter. *IEEE Journal of Solid-State Circuits*, 27(3):351–358, 1992.

[21] Dave Treleaven. 1.5-bit stages in pipelined adc, May 2006.

[22] Rudy van de Plassche. *CMOS Integrated Analog-to-Digital and Digital-to-Analog Converter*. Second edition, 2003.

[23] Li Zhang. The calibration technique for pipelined adc. *2008 International Conference on MultiMedia and Information Technology*, 2008.