Signal ADC Converter Simulation on Cadence Virtuoso for Audio Applications

by

Maxwell Kazuki Fukada

Senior Project
Electrical Engineering Department
California Polytechnic State University
San Luis Obispo
June 2022
## Contents

Contents .................................................................................................................................................. 2  
List of Figures ......................................................................................................................................... i  
List of Tables ......................................................................................................................................... ii  
Abstract ................................................................................................................................................ iii  
Chapter 1: Introduction ......................................................................................................................... 1  
Chapter 2: Background .......................................................................................................................... 2  
Chapter 3: Requirements ....................................................................................................................... 7  
Chapter 4: Design ................................................................................................................................... 10  
Chapter 5: Development ....................................................................................................................... 14  
Chapter 6: Testing and Verification ....................................................................................................... 18  
Chapter 7: Conclusion ........................................................................................................................... 25  
Acknowledgements ............................................................................................................................... 28  
References ............................................................................................................................................... 29  
Appendix A: Senior Project Analysis ..................................................................................................... 30  
Appendix B: Verilog-A Programming Found in Components ............................................................... 33
List of Figures
Figure 1.1: ADC architectures, applications, resolution, and sampling rates [1] ................................................. 1
Figure 2.1: Effect of Oversampling Quantization Noise ......................................................................................... 2
Figure 2.2: Example Sigma-Delta Modulator Output Waveform ............................................................................ 3
Figure 2.3: First-Order Sigma-Delta Modulator Structure [3] ................................................................................. 4
Figure 2.4: Block Diagram of Sigma-Delta Modulator .......................................................................................... 4
Figure 2.5: Example Waveforms of the Circuit in Figure 2.4 ................................................................................. 5
Figure 2.6: Effects of Digital Filtering and RC filter ............................................................................................... 5
Figure 2.7: Decimation Process’s Input and Output ............................................................................................... 6
Figure 3.1: Level 0 Decomposition of Converter ................................................................................................ 7
Figure 3.2: Level 1 Decomposition of Converter ................................................................................................ 8
Figure 3.3: How the Frequency of the Signal Data Changes ..................................................................................... 9
Figure 4.1: Passive RC Low Pass Filter Circuit ..................................................................................................... 10
Figure 4.2: Final RC Low Pass Filter Design ......................................................................................................... 11
Figure 4.3: Planned Sample and Hold Design ........................................................................................................ 11
Figure 4.4: Final First-Order Sigma-Delta Modulator Design .................................................................................. 12
Figure 4.5: Decimation Module Block Diagram .................................................................................................. 13
Figure 5.1: Implementation of a passive Analog LP Filter on Virtuoso ................................................................. 14
Figure 5.2: Implementation of Sample and Hold Block on Virtuoso .................................................................... 14
Figure 5.3: SampHold1 Object Parameter Setting ............................................................................................... 15
Figure 5.4: Implementation of Σ-Δ Modulator on Virtuoso ................................................................................... 15
Figure 5.5: Implementation of Decimation Block in Virtuoso ................................................................................. 16
Figure 5.6: Logic Diagram of ClockCounter ...................................................................................................... 16
Figure 5.7: Logic Diagram of SimpleAveraging ................................................................................................... 16
Figure 5.8: Complete Schematic of the Converter System ..................................................................................... 17
Figure 6.1: Simulation Result of RC- Low Pass Filter .......................................................................................... 18
Figure 6.2: Waveform Results of a Simulation of Sample and Hold Block .......................................................... 19
Figure 6.3: Simulation Result of Σ-Δ with 5VDC as Vin ...................................................................................... 19
Figure 6.4: Simulation Result of Σ-Δ with -5VDC as Vin ..................................................................................... 20
Figure 6.5: Simulation Result of Σ-Δ with 4kHz Sinusoidal, 5V Amplitude Vin .................................................... 20
Figure 6.6: Simulation Result of Σ-Δ with 40kHz Sinusoidal, 5V Amplitude Vin ..................................................... 21
Figure 6.7: Simulation Result of System with 5VDC as Vin .................................................................................. 22
Figure 6.8: Simulation Result of System with 0VDC as Vin .................................................................................. 22
Figure 6.9: Simulation Result of System with -5VDC as Vin ................................................................................ 23
Figure 6.10: Simulation Result of System with 4kHz Sinusoidal, 5V Amplitude Vin ............................................. 23
Figure 6.11: Simulation Result of System with 40kHz Sinusoidal, 5V Amplitude Vin .......................................... 24
Figure 7.1: Initial Implementation of Sample and Hold Block on Virtuoso ............................................................ 25
Figure 7.2: Design of CMOS_Inverter ................................................................................................................ 26
Figure 7.3: Initial Design of D-Flipflop .................................................................................................................. 26
Figure 7.4: Second-Order Sigma-Delta Modulator Design ................................................................................... 27
List of Tables

Table 3.1: I/O of Converter.................................................................................................................. 7
Table 3.2: I/O and Functionality of Analog Low Pass Filter ................................................................. 8
Table 3.3: I/O and Functionality of Sample and Hold............................................................................ 8
Table 3.4: I/O and Functionality of Σ-Δ Modulator.............................................................................. 9
Table 3.5: I/O and Functionality of Decimation .................................................................................... 9
Table 5.1: Cadence Ideal Components Used in ADC.......................................................................... 15
Abstract

Audio signals are representations of sounds with a mixture of multiple analog signals between the frequency of 20Hz to 20,000Hz. To record snippets of audio data onto a mobile phone or computer, the signal needs to be converted to a digital format. For this purpose, many devices utilize a converter, specifically a sigma-delta modulator with a digital filter. By using a converter, electronics can receive binary data about the audio signal accurately and quickly without losing important signal information. This project aims to simulate a fully functional audio converter with a sigma-delta modulator and decimation filter. The system will receive any analog signal between 20Hz and 22.05kHz while being oversampled at 2.8224MHz with a 64 oversampling ratio value, meaning the sampling frequency is 44.1kHz. As a result, the system will be outputting a corresponding digital 6-bit binary value at the rate of 44.1kHz. This whole system will be running on a +/- 5V power supply. While this project concentrates on the simulation of the device using some real components and Verilog-A, the programmed components can be replaced with real electronic components to create a fully operational VLSI converter design.
Chapter 1 : Introduction

In simple terms, an audio converter is an ADC attached to a digital filter. In the industry, there are multiple types of ADCs, and each type has its advantages and disadvantages. Three main ADC types represent a very large percentage of the available in today’s market: Successive Approximation (SAR), Delta-Sigma or Sigma-Delta ($\Sigma-\Delta$), and Pipeline [1].

As shown in Figure 1.1, each ADC has its own capability of sampling rate (horizontal axis) and resolution (vertical axis). SAR is the most popular, especially when multiple channel inputs, also known as multiplexing, are required [1]. SAR is known to have a simple circuit and has the capability of handling higher sampling rates, but with higher sampling rates, it experiences limitations in bit resolution and dynamic range [2]. When a higher sampling rate is needed, Pipeline ADCs are usually used. While it’s known to have high speeds, it is even more limited by its bit resolution compared to the SAR [2]. $\Sigma-\Delta$ is used when the bit resolution is valued more than the sampling rate or the circuit speed. The design of $\Sigma-\Delta$ also makes them ideal for dynamic applications. $\Sigma-\Delta$ works by over-sampling the signal much higher than the targeted sampling rate. This will allow the DSP to create a higher-resolution data stream which then allows for a multistage anti-aliasing filter. With anti-aliasing filters, there will be no signals above the Nyquist frequency, making it almost impossible to digitize false information [2].

This natural anti-aliasing property allowed $\Sigma-\Delta$ to dominate modern audio applications. $\Sigma-\Delta$ ADCs could be seen in FM stereos, computers, audio CDs, digital audio tape, and DVD audio [1]. Modern-digital cellular systems use higher-resolution oversampled linear of $\Sigma-\Delta$ ADCs and DACs [1]. These $\Sigma-\Delta$ are incorporated into CODECs (data coders and decoders) which are applications used in pulse code modulators, speech processing, encryption, etc. [1].

This report explains the background of how a Sigma-Delta converter works and the concepts behind the system. The report has detailed notes on how to design these types of converters and what to expect when the system is being simulated and verified.

![Figure 1.1: ADC architectures, applications, resolution, and sampling rates [1]](image)
Chapter 2: Background

2.1 Oversampling
As mentioned previously, Σ-Δ utilizes a much faster sampling rate than the targeted sampling rate. By having oversampling, the sampled signal’s noise is spread throughout a wider spectrum of frequencies.

As seen in Figure 2.1, with a higher sampling rate, the Nyquist frequency also increases as an effect. Even though the Nyquist frequency increased, the total quantization noise will remain the same. This will allow the quantization noise to spread out between the frequencies of 0 and Kf_s/2, K being the oversampling ratio. Oversampling also allows the increase of the overall signal-to-noise (SNR) ratio by 10log_{10}(K) [3]. While this means that the SNR can be easily manipulated by increasing the oversampling ratio, it is important to keep in mind that the oversampling ratio needs to be increased by 4 to increase the SNR by only 6dB.

Another positive effect observed is that the actual desired signal stays at the same location of the frequency spectrum, so relaxing the roll-off requirement of the analog antialiasing filter will cause no loss of signal information. The initial analog input filter can be designed to start stopping frequencies about Kf_s/2.”
2.2 Basics of Sigma-Delta Modulators

The main goal of a $\Sigma$-$\Delta$ modulator is to convert the input signal into a continuous stream of HIGHs and LOWs at the rate of the clock that is connected to it.

![Sigma-Delta Modulator Waveforms](image)

For example, as seen in Figure 2.2, when the modulator observes the maximum input value, it will output a continuous data stream of HIGHs. Vice versa, when it observed the minimum value, it will output a data stream of LOWs. When the modulator receives a signal that is half of the maximum, it will output a data stream alternating between HIGHs and LOWs every clock cycle. Anything above the center will output HIGHs more often than LOWs over several clock cycles. An important thing to note is that observing a single output from one clock cycle will be not useful information. The output needs to be observed over multiple clock cycles and averaged depending on how many HIGHs there are over a certain number of clock cycles. Using Figure 2.2 again, if the middle output waveform is observed over 8 outputs, it can be seen that there are 4 HIGHs of 8 outputs or an average of ½. This follows the correlation that the input is half of the max voltage. If the input was slightly higher, the number of HIGH over 8 outputs would now be 5 instead of 4.
The basic structure of the Σ-Δ is introduced in Figure 2.3, but since it is hard to visualize the operation of the whole system, a simpler block diagram is introduced in the Figure 2.4:

The four main components involved in the Sigma-Delta Modulator are the summation block, integrator, comparator, and impulse block. With these four components working together, the system outputs a stream of HIGHs and LOWs with different ratios depending on the input value. An example of the possible waveforms is shown in Figure 2.5:
In Figure 2.5, it shows an example system running on +1V and -1V. The threshold value of the comparator is 0V as shown in the Node B waveform. As seen in the figure, all waveforms except the output values follow some pattern with the CLK. Since the impulse block is connected to the CLK, it will only update its value at the rising edge of the CLK, so even if the actual output value is updated to a HIGH, the impulse block will wait until the next rising edge to change its value to HIGH. This causes a domino effect for the summation block and integrator to also wait until the next rising edge to have any change in the waveform. This explains why the integrator continues to rise or fall even after it crosses the threshold value.

### 2.3 Digital Filtering and Decimation

![Figure 2.6: Effects of Digital Filtering and RC filter](image)

Figure 2.6 describes the purpose of the two filters that are seen in a typical Σ-Δ converter. While the analog RC filter takes care of all frequencies above the Nyquist frequency of the
oversampling frequency, another digital filter is needed to filter the frequencies above the targeted sampling frequency. This is done using a digital filter after the analog input signal is converted into digital. Both finite impulse filter (FIR) and infinite impulse filter (IIR) are used in these cases, but FIR filters are often used in audio applications because they are always stable and keep linear phase characteristics [3]. Another purpose the FIR serves is to decimate the oversampled input signal to throw away unnecessary information. The sampled frequency is reduced by a factor of the oversampling ratio and the resulting output signal has still enough information to accurately represent what the input signal was [5]. This process could also be described as an averaging filter since it takes the average of many samples to describe the value seen around that time.

As seen in Figure 2.7, the output-data rate has significantly fewer samples than the input, but it still retains the general sinusoidal wave. It is important to note that the output data is still represented in binary. The binary code in the vertical axis represents the corresponding analog value. For example, the system can be designed to represent FFFFFF as maximum analog input and 000000 as minimum analog input.
Chapter 3: Requirements

3.1 Level 0 and Level 1 Decomposition

The level 0 and level 1 decomposition figures (Figures 3.1 and 3.2) below show the overview of the Audio Converter design. The Level 0 Decomposition details the overall inputs and outputs of the system while the Level 1 decomposition shows a summarization of the key components within the converter.

As seen in Figure 3.1, the Converter has 5 inputs and 1 bus output. The overall goal of the system is to compress the analog input audio signal to an unsigned 6-bit digital signal that represents the magnitude of the received analog input. All components will be powered by a +/-5V_{DC} power supply while mainly running on a 2.8224 MHz clock. Another clock running at 282.24 kHz for a slower running sample and hold component would be needed. It should be noted that the slower clock would be designed to be produced internally in the system in real-life applications, but for this simulation, there would be two different clock inputs.

<table>
<thead>
<tr>
<th>Module</th>
<th>Converter System</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inputs</strong></td>
<td></td>
</tr>
<tr>
<td>Audio Input Signal:</td>
<td></td>
</tr>
<tr>
<td>o Voltage: between -5V_{min} and 5 V_{max}</td>
<td></td>
</tr>
<tr>
<td>Clock Signal: 2.8224 MHz</td>
<td></td>
</tr>
<tr>
<td>Clock for S+H: 282.24 kHz</td>
<td></td>
</tr>
<tr>
<td>Power Supply: +5V_{DC} and -5V_{DC}</td>
<td></td>
</tr>
<tr>
<td><strong>Outputs</strong></td>
<td></td>
</tr>
<tr>
<td>Audio Output Signal: represented in 6bits, 6'b000000 for min and 6'b111111 for max values. Each bit will have 5V_{DC} for HIGH and 0V_{DC} for LOW</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: I/O of Converter
The internals of the converter are shown in Figure 3.2. There are four different components that work together to create this converter: Analog Low Pass Filter, Sample and Hold, Sigma-Delta Modulator, and Decimation. The main audio input will be between -5V and +5V when it goes into the system. The audio signal will be kept between this voltage until it is processed through the Sigma-Delta modulator as the quantized signal. After the Σ-Δ modulator, the voltage will be either 2.5V or -2.5V. Once it goes through the decimation filter and output of the system, each bit will be represented in either 5V as HIGH or 0V as LOW. Each individual blocks detailed I/O and the functionality are described through Tables 3.2 and 3.5.

### Table 3.2: I/O and Functionality of Analog Low Pass Filter

<table>
<thead>
<tr>
<th>Module</th>
<th>Analog Low Pass Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inputs</strong></td>
<td>Audio Input Signal: Between -5V_{min} ~ 5 V_{max}, any frequency</td>
</tr>
<tr>
<td><strong>Outputs</strong></td>
<td>Filtered Input: Between -5V_{min} ~ 5 V_{max}, all frequencies under 44.1kHz</td>
</tr>
<tr>
<td><strong>Functionality</strong></td>
<td>The low pass filter is used to filter any of the original analog signal’s high frequencies that are larger than the targeted frequency spectrum’s maximum.</td>
</tr>
</tbody>
</table>

### Table 3.3: I/O and Functionality of Sample and Hold

<table>
<thead>
<tr>
<th>Module</th>
<th>Sample and Hold</th>
</tr>
</thead>
</table>
| **Inputs** | Filtered Input: Between -5V_{min} ~ 5 V_{max}, all frequencies under 44.1kHz  
Clock for S+H: 282.24 kHz  
Power Supply: +5V_{DC} and -5V_{DC} |
| **Outputs** | Sampled Signal: Between -5V_{min} ~ 5 V_{max}, all frequencies under 44.1kHz |
| **Functionality** | The Sample and Hold component are used to hold a particular value for a longer period time than the Σ-Δ modulator. This will allow the Σ-Δ modulator to analyze the input signal multiple times while keeping its stability. |
### Module

<table>
<thead>
<tr>
<th>Module</th>
<th>Sigma-Delta (Σ-Δ) Modulator</th>
</tr>
</thead>
</table>

### Inputs

- **Sampled Signal:** Between $-5V_{\text{min}} \sim 5V_{\text{max}}$, all frequencies under 44.1kHz
- **Power Supply:** $+5V_{\text{DC}}$ and $-5V_{\text{DC}}$
- **Clock Signal:** 2.8224 MHz

### Outputs

- **Quantized Signal:** $2.5V_{\text{DC}}$ HIGH, $-2.5V_{\text{DC}}$ LOW

### Functionality

Receives the sampled analog signal and converts the information to a data stream of multiple HIGHs and LOWs. Depending on the amplitude of the analog signal, the ratio of HIGHs and LOWs will change.

**Table 3.4: I/O and Functionality of Σ-Δ Modulator**

### Module

<table>
<thead>
<tr>
<th>Module</th>
<th>Decimation</th>
</tr>
</thead>
</table>

### Inputs

- **Quantized Signal:** $2.5V_{\text{DC}}$ HIGH, $-2.5V_{\text{DC}}$ LOW
- **Power Supply:** $+5V_{\text{DC}}$ and $-5V_{\text{DC}}$
- **Clock Signal:** 2.8224 MHz

### Outputs

- **Audio Output Signal:** represented in 6 bit, $6'b000000$ for min and $6'b111111$ for max values. Each bit will have $5V_{\text{DC}}$ for HIGH and $0V_{\text{DC}}$ for LOW

### Functionality

Will observe the number of HIGHs and LOWs received from the Σ-Δ modulator over a certain number of clock cycles. It will add the number of HIGHs observed and output that in binary form using its 6-bit output.

**Table 3.5: I/O and Functionality of Decimation**

### 3.2 Frequency of the Signal

In Figure 3.3, it shows how the frequency changes as the signal progress through the system. The signal will only be increased after it passes through the Σ-Δ. In this case, the desired $f_s$ would be 44.1kHz since humans can at most only hear between the range of 20Hz and 2000Hz [4] and 44.1kHz is the frequency commonly used in CDs. The K, the oversampling ratio, will be 64. This follows the convention of needing $2^N$ oversampling ratio to have an output of N bits [3]. In this case, since 6 bits are targeted, $2^6$ or 64 is chosen as the oversampling ratio.
Chapter 4 : Design

4.1 Analog Low Pass Filter

As discussed earlier, since the Σ-Δ has natural anti-aliasing properties, a high-order active low pass filter does not need to be implemented. A simple RC low-pass filter, like shown in Figure 4.1, will be sufficient to filter all the unnecessary high frequencies from the input signal.

The cutoff frequency that is being targeted would be the Nyquist frequency of the oversampling rate’s frequency. In this system, the sampling frequency, $f_s$, is 44.1kHz and the oversampling ratio, $K$, is 64.

$$f_{os} = Kf_s = (64)(44.1kHz) = 2.8224MHz$$

Since the oversampling frequency is 2.8224MHz, our Nyquist frequency will be:

$$f_{os}/2 = 1.4112MHz$$

The cutoff frequency of a simple passive RC low-pass filter can be found as followed:

$$f_c = \frac{1}{2\pi RC} = 1.4112MHz$$

It is desirable to have a capacitance than a lower resistance in VLSI since capacitors are much larger, a much lower capacitor value was chosen compared to the resistor value. If 11.277kΩ was chosen for $R$ and 1pF for $C$, the cutoff frequency will be:

$$\frac{1}{2\pi RC} = \frac{1}{2\pi(11.277k\Omega)(1pF)} = 1.4113MHz = f_c$$

The final circuit design of the passive RC-filter is shown in Figure 4.2:
4.2 Sample and Hold

The Sample and Hold component’s main goal is to retain the value over a period. This could be easily achieved by switching a switch and a capacitor large enough to keep the input value’s charge and discharge it long enough, but further modifications can be implemented to improve the value sampled and outputted.

As seen in Figure 4.3, the designed sample and hold block has many additional components apart from a switch and capacitor. The two amplifiers in the input and outputs are configured to be buffers. These buffers will compensate for the low drop-out voltage across the holding capacitor. Also, an inverter and PMOS were added additionally with the NMOS to ensure that the internal channel capacitance of the transistors cancels each other. The capacitance of the capacitor can be a wide range of values. Any value can be used if the capacitor is able to charge itself fast enough to keep up with changing input values and sustain a stable value when outputting values.

4.3 ADC

The basic design of the Σ-Δ was shown in Chapter 2 and that will be incorporated into the actual circuit that was built. The impulse block can be easily fulfilled with a D-flipflop since it
essentially does the same function since it follows the CLK timing and outputs only either a HIGH or LOW. The summation block was changed to a differential amplifier which will also do the same function as a summation block. An active RC integrator was designed as an integrator. To understand how the resistor and capacitor values for the integrator were chosen, the waveform of Node B from Figure 2.4 should be observed. An ideal integrator will always attempt to have a linear slope to constant input. For an integrator, a linear response is observed earlier in the time constant, $\tau$. If the CLK frequency is used as a reference for how long an integrator may experience a constant input, the desired $\tau$ can be calculated.

$$T_{os} = \text{Period of } CLK = \frac{1}{f_{os}} = \frac{1}{2.8226\text{MHz}} = 0.3543\mu s$$  \hspace{1cm} (4.5)

In a lot of cases, to see a linear characteristic response in the integrator, $\tau$ should be more than 10 times larger than the desired period. In this design, a time constant at least 10 times larger than the CLK was selected.

$$\tau \gg (10)T_{os} = (10)(0.3543\mu s) = 3.543\mu s$$ \hspace{1cm} (4.6)

The RC values can be chosen in which the product will be the desired time constant. In this design, the resistor value was chosen to be 153k$\Omega$ and the capacitor value to be 23.3nF.

$$\tau = 3.556\mu s = RC = (153k\Omega)(23.3nF)$$ \hspace{1cm} (4.7)

When designing the passive RC filter, it was discussed that smaller capacitor values were desirable since they take up more space compared to resistors. In this case, since increasing capacitors had a better performance in the simulation, the larger capacitance was chosen. In real-life applications, this huge capacitance will be obtained with an external capacitor.

In the complete ADC circuit shown in Figure 4.4, it can be observed that the output data is taken from a NOR gate with the inputs from the $\bar{Q}$ value and the CLK. It was seen on Figure 2.5 that the output from the comparator did not follow the CLK cycles if the design from Figure 2.3 was
used. If the output isn’t following the CLK edges, it makes the signal hard to manipulate in the
decimation section of the converter, so by taking the $Q$ value and comparing it to the CLK in the
NOR gate, the output value will follow CLK cycles while keeping its change in HIGHs and
LOWs.

4.4 Decimation
While a digital filter and decimation module are placed in this section by using an FIR filter, this
process is complicated and has many restrictions that need to be followed. For simulation
purposes, it is assumed that the received signal from the $\Sigma$-$\Delta$ has no quantization noise below the
frequencies of the Nyquist frequency of the sampling frequency ($f_s/2$). Because there is no noise,
the decimation module only needs to take in multiple samples over multiple clock cycles and
convert the average value into a binary format. This could be done by two blocks shown in
Figure 4.5.

![Decimation Module Block Diagram](image)

The ClockCounter block in Figure 4.5 will count the number of CLK cycles that pass through the
system. Once it reaches the oversampling ratio, $K$, it will send an interrupt signal to the other
block. In this case, since the oversampling ratio is 64, it will send an interrupt after counting to
64. The other block, SimpleAveraging, will observe the HIGHs and LOWs data stream from the
$\Sigma$-$\Delta$ modulator. The block will count how many HIGHs are being sent from the $\Sigma$-$\Delta$. At the
same time, it is waiting for an interrupt signal from ClockCounter. Once an interrupt signal is
received, the SimpleAveraging block will output the number of HIGHs it has counted and
convert that value into a binary format.

This chapter described the design and considerations that were taken before anything was built
on Cadence Virtuoso. Clear component values and schematics were designed, so the circuits can
be implemented onto Cadence Virtuoso in the next chapter.
Chapter 5 : Development

5.1 Analog Low Pass Filter

Figure 5.1: Implementation of a passive Analog LP Filter on Virtuoso

While the actual resistor and capacitor values are not shown in Figure 4.2, the exact value described in Figure 5.1 was implemented.

5.2 Sample and Hold

Figure 5.2: Implementation of Sample and Hold Block on Virtuoso

The original design in shown in Figure 4.3 was attempted, but because of issues caused in the original designs as explained later in Chapter 7: Conclusion, the ideal Sample and Hold module found in Cadence’s library was used instead. In Cadence, there are pre-made components that are created using a program called Verilog-A. These components are found in multiple libraries that are also provided by Cadence which can be used in circuits that are being simulated. It is also important to note that anyone can program functions in Verilog-A in the Cadence environment to create their own components that will run on the circuit. This particular component is called SampHold1, found in the bmslib library. The object’s parameters were set as shown in Figure 5.3.
5.3 ADC

For some components used, the ideal components given by one of Cadence’s libraries were used. Table 5.1 provides the used cell’s name and which libraries they were found in:

<table>
<thead>
<tr>
<th>Purpose Relative to Figure 4.4</th>
<th>Cell Name</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differential Amplifier</td>
<td>limiting_diffamp</td>
<td>ahdLib</td>
</tr>
<tr>
<td>Op Amp</td>
<td>limiting_diffamp</td>
<td>ahdLib</td>
</tr>
<tr>
<td>D-Flipflop</td>
<td>d_ff</td>
<td>ahdLib</td>
</tr>
<tr>
<td>NOR Gate</td>
<td>nor_gate</td>
<td>ahdLib</td>
</tr>
</tbody>
</table>

Table 5.1: Cadence Ideal Components Used in ADC

Since all of these components in this library are programmed using Verilog-A, the components work as desired as long as they are coded properly. The comparator was also built off Verilog-A and the original code can be found in Appendix B. The RC values are not shown in Figure 5.4, but they exactly follow the values that were shown in Figure 4.4.
5.4 Digital Filter and Decimation

Both blocks shown in Figure 5.5 were implemented by programming them in Verilog-A. The actual code can be found in the Appendix B, but the logical diagram will be shown.

As seen in Figure 5.6, the logic counts each rising edge of the CLK input and will cause an interrupt signal to go HIGH as soon as it counts 64 CLK cycles. This would allow the system to keep track of the oversampling rate compared to the CLK signal.

As seen in Figure 5.7, the logic counts each rising edge from the data stream received from the Σ-Δ modulator. Once it receives the interrupt signal from ClockCounter, it will transfer the number of HIGHs from Σ-Δ to an equivalent 6-bit number and outputs each individual bit.
Figure 5.8 shows the complete circuit design that was used for the system simulation. In the SimpleAveraging object in the decimation stage, there is an extra output pin named “onecounts”. This is a pin that will output the result of the HIGHs counted by the system in decimal form rather than binary form. This pin is only for testing and verification purposes. If a manufacturer was planning to build this design’s IC, the pin will be deleted.

In this chapter, it was shown how all the components that were designed in Chapter 4 were implemented onto Virtuoso. It was shown how all designs that were initially planned didn’t always work and some alternative designs had to be implemented. In the next chapter, it will observe and analyze the results from the circuits that were build in this chapter.
Chapter 6 : Testing and Verification

6.1 Analog Low Pass Filter

By inputting a 1V voltage and seeing the frequency response of the implemented circuit in Figure 5.1, it can be verified if the filter is successfully able to filter any frequency above $f_c$ or 1.4112MHz as shown in Equation 4.3.

![Figure 6.1: Simulation Result of RC- Low Pass Filter](image)

As seen in Figure 6.1, the filter begins to filter out frequencies near the desired cutoff range. The cutoff frequency is described as the frequency when the input signal’s magnitude is reduced by -3dB or $\sqrt{\frac{1}{2}}$ in linear terms. In this case, since the input voltage is 1V, the cutoff frequency would be 0.707V. At 1.411MHz or the desired cutoff frequency, the magnitude is 0.70826 which confirms that the design works properly and effectively.

6.2 Sample and Hold

While the Sample and Hold block is ideal and programmed in Verilog-A, it was verified that it met the specification requirements that were set in the beginning.
The Sample and Hold Clock, S_H_input, will have the frequency of 282.24kHz or the tenth of the oversampling frequency. 282.24kHz converts to 3.56µs periods, so a pulse waveform with the amplitude of +/- 2.5V with 50% duty cycle with a period of 3.56 µs was inputted, as shown in the Figure 6.2 as the purple waveform. A sinusoidal waveform with a 5V amplitude and 40kHz frequency was sent as an input voltage (green waveform). In Figure 6.2, the output waveform (red waveform) shows it perfectly changing its value to the current input voltage value when the S_H_input is at its rising edge. It also perfectly retains its value while the S_H_input is not at its rising edge. It is confirmed that the SampHold1 works as desired and meets the specification requirements.

6.3 ADC
Verification and comprehension of the ADC is more complicated since there are many parts involved in this circuit.
Figure 6.4: Simulation Result of ΣΔ with -5VDC as Vin

Figure 6.3 and 6.4 both show results of the simulation when the inputs are at their extreme, the maximum and minimum value, respectively. For Figure 6.3, it can be confirmed that the output, $V_{\text{out}}$, has a HIGH for every period of the CLK. This will result in the counting a HIGH for every clock cycle in the decimation module. There are no changing waveforms in any other components of the ADC because there is no changing input signal. Vice versa, Figure 6.4 shows that $V_{\text{out}}$ is always LOW. This will result in a count of 0 in the decimation module. For the same reason, there are no changing waveforms in any other components since the input is not changing.

Figure 6.5: Simulation Result of ΣΔ with 4kHz Sinusoidal, 5V Amplitude Vin

Once changing signals are introduced, the waveforms become more complicated. Figure 6.5 shows when the input signal is a sinusoidal wave with a 5V amplitude and a 4kHz frequency. It can be observed that as VIN reaches its maximum peak, the number of HIGH at the VOUT is also at its max. As VIN decreases in value, so does the number of HIGH. Once VIN reaches its minimum peak, VOUT continuously outputs zero for that period. The waveform at the output of
the integrator, $V_{integrator test}$, is observed, and it is confirmed that the signal oscillates between about -500mV and 500mV. The signal will continue to rise when the diffout signal is LOW, but as soon as it crosses the threshold of 0V and the next CLK cycle comes, the signal will fall because the diffout signal is now HIGH.

![Figure 6.6: Simulation Result of $\Sigma-\Delta$ with 40kHz Sinusoidal, 5V Amplitude Vin](image)

Figure 6.6 is similar to Figure 6.5. The input signal is sinusoidal and has the same amplitude, but the frequency is changed to 40kHz. Even after an increase in input frequency, the $V_{OUT}$ retains its ADC pattern and outputs a high rate of HIGHs at the maximum peak while outputting LOWs at the minimum peak.

6.4 Converter System

The decimation module is hard to confirm without the use of the ADC modulator already connected to it. The next results are simulation results of the whole system, as shown in Figure 5.8.

The figure below shows the results of the decimation module when the input signal into the ADC has the highest possible input of $5V_{DC}$. In the ADC module, the components are operating as seen in Figure 6.3.
Since the input value is at its maximum, the expected value of the binary output should also be at its maximum. This expectation can be confirmed by observing the waveforms b5, b4, b3, b2, b1, and b0 in Figure 6.7. All bits are HIGH, signifying the highest possible binary value. This value continues throughout the whole simulation of 1ms, so it can be said that the system is stable and outputs the same value continuously throughout multiple CLK cycles. By observing the interrupt signal and the onecounts pin, it can be confirmed that every time the interrupt signal is HIGH, the onecounts pin resets its value to 0. This is what was expected and intentionally done in the development section.

Figure 6.8 shows when the input value is $0V_{DC}$ or exactly the center of the maximum and minimum input values. The expected output binary value would be half of the maximum binary value. Since the maximum binary value is 63, either value of 32 or 31 would be expected. B5 is
LOW while all other bits are HIGH, this will translate to 011111 or 31 in decimal form. This confirms the expected results and confirms that the system is working properly for this input.

Figure 6.9 is like Figures 6.7 and 6.8, but the input voltage is at the minimum value. At the minimum input, the output values are also at their minimum value of 000000.

Figure 6.10 shows the output values when the input signal is a changing signal. The situation in the ADC module is the same as in Figure 6.5. Since the value is changing over a large time period compared to the CLK period, the average values over 64 CLK cycles change depending on what part of the 4kHz sinusoidal wave the input signal is at. The effect of this can be observed in the onecounts waveform. As the input value is near its maximum value areas, the counted HIGHs are much higher than when the input values are near their minimum value areas. This signifies that the ADC is correctly changing the ratio between the HIGHs and LOWs in its output data stream as the input amplitude changes. This also signifies that the decimation
module is correctly averaging 64 CLK cycles and outputting the counted ones in that time period.

Figure 6.11: Simulation Result of System with 40kHz Sinusoidal, 5V Amplitude Vin

Figure 6.11 is like Figure 6.10, but the frequency of the input signal is now 40kHz. It can be observed much less change in the onecounts waveform. This is because the time period for the changing input values is much closer to the CLK time period, compared to a 4kHz input signal. This also means that when averaging 64 CLK cycles, the input signal has gone through much more of its change. This change will mean that the difference between the averages of the signal would be similar in value compared to each other.
Chapter 7: Conclusion

The original goal of this project was to construct and simulate a working signal converter for audio applications. The results shown in Chapter 6 show that the final design is working as intended and successfully converting the input analog signal into the corresponding 6-bit binary output. The final design will be able to handle any input signal between -5V and 5V with a signal frequency between 20Hz and 22.05kHz since it is being sampled at 44.1kHz with oversampling ratio of 64. The whole system will run with a power supply of +/-5V.

7.1 Restrictions Experienced in Process

In the original design, the whole system was planned to run on between 0V and 5V. The change of introducing a negative power supply was done after finding multiple problems found in the existing op-amp module and available Cadence provided Verilog-A programmed objects. By using only positive voltages, the design and use of CMOS logical gates were allowed. CMOS designs were used for Sample and Hold and D-flip-flop blocks.

Figure 7.1: Initial Implementation of Sample and Hold Block on Virtuoso

Figure 7.1 shows a successful implementation of the Sample and Hold circuit design based on the design shown in Figure 4.3. The issue popped up once negative input and CLK voltages were introduced. Comparator_mcarro05_F20 is an operational amplifier provided by Professor Smilkstein, but since it is not an ideal component there were many limitations that affected the results. The operational amplifier could not operate from rail to rail of the supply voltage.
Another issue was caused by the CMOS inverter. Initially a CMOS inverter, like shown in Figure 7.2 was built. Since it is a CMOS inverter, it was not able to invert a negative input value to the corresponding output value. These issues and restrictions caused the design to implement an ideal Sample and Hold block found in one of Cadence’s libraries that are coded in Verilog-A.
Figure 7.3 also shows the initial design of the D-flipflop made from CMOS transistors. The design was scratched after negative voltages were introduced for the same reason as the Sample and Hold module.

7.2 Future Improvements
As seen in the design phase, there were Verilog-A components used in this simulation. By using Verilog-A programmed modules, it makes simulation much easier since it doesn’t introduce any power loss or noise. It is valuable and shows proof of concept. When trying to make a working product, a physical design will be needed for many of these components. The most important component would be an operational amplifier. With a correctly designed operational amplifier, it can be used to replace the ideal differential amplifier, the integrator, and the comparator in the Σ-Δ modulator. Also, once this is corrected, negative voltages would not have to be necessary since the reason why negative input voltages were introduced is because of the ideal components. No negative voltages would allow the design to use CMOS logical gates in which the designs introduced in Figures 7.2 and 7.3 could be back into use for the actual component.

Another improvement the system can go through is increasing the order of the filter in the Σ-Δ modulator. Just like other analog filters, higher order filters will allow the Σ-Δ modulator to perform better. With a higher order, more of the shaped quantization noise, as shown in Figure 2.1, is pushed further to higher frequencies. This allows for the resulting output’s SNR to be better [3]. In the final design, a second order Σ-Δ will be designed as shown in Figure 7.4.

![Second-Order Sigma-Delta Modulator Design](image)

When a first-order modulator is used and oversampled by 64, an SNR of about 40 can be expected. If a second-order modulator is used instead, an SNR of about 75 can be expected [3]. This increase in SNR signifies that the output resolution can be as high as 13 bits.

The other improvement that should be considered is introducing a digital filter. As more real-life components are introduced, the noise will become more prevalent and the need for a digital filter would become necessary. A digital filter that could be implementable in the Cadence program environment Cascaded Integrator Comb (CIC). CICs are suited for anti-aliasing filtering before decimation, so they are often used in Σ-Δ A/D converters in the industry [6]. The topic of CIC itself is extremely complicated and could be a whole other project by itself.
Acknowledgements

I would like to thank Professor William Ahlgren for guiding me through this Senior Project and providing tips on how to successfully finish this project.

I would also like to give a very special thanks to Professor Tina Smilkstein for countless hours of Zoom meetings and providing me tips on how to use the Cadence software. Without Tina’s critical observations and advice, I would have not been able to successfully have a working simulation at the end.
References


Appendix A: Senior Project Analysis

**Project Title:** Signal ADC Converter Simulation on Cadence Virtuoso for Audio Applications

**Student’s Name:** Maxwell Kazuki Fukada

**Advisor’s Name:** William L. Ahlgren

**Summary of Functional Requirements:**
The Signal Converter receives an audio or analog input signal between -5V and 5V with a frequency up to 22.05kHz since it is being oversampled by 64 of the sampling frequency of 44.1kHz. The design will convert the signal to the corresponding 6-bit digital binary code based on a Sigma-Delta ADC modulator. The resulting data can be used for the storage of audio data in electronic devices such as computers, mobile phones, etc.

**Primary Constraints:**
Many of the components used in the design were ideal components created by Verilog-A program in the Virtuoso environment. Initially, real components made with transistors were used but were scrapped because of various issues. These issues included restrictions in output voltage, inaccurate transient responses, incapability of handling negative voltages, and too much noise. Another constraint was the time given. In the IC industry, it takes many months or maybe years to perfect a chip. Finishing everything in 9 months was not long enough to add and experiment with everything possible. Certain requirements had to be prioritized over others to meet the deadline.

**Economic:**
- I. Human Capital: The converter will help its surrounding companies that are manufacturing this device which will as a result help create more jobs in sectors like engineering, marketing, and manufacturing.
- II. Financial Capital: Customers will be able to compress audio signals with one single chip while currently.
- III. Natural Capital: This product will be heavily reliant on rare materials like copper, gold, and silicon. These materials can be renewable if done properly, so it is important to note that in the datasheet or the marketing.
- IV. Cost: Because the product is still in the designing phase, the cost of fabricating the IC can change depending on the number of layers, area, complexity, etc. Currently, since all experimenting and testing were done in the Cadence Virtuoso environment, the total cost used is $0. As the design becomes finalized and real-life components are used, plans for manufacturing an actual physical IC test chip would occur. Custom chips do cost a lot and manufacturers may charge thousands of dollars for a couple of test dies. Once the IC
is received, electrical measuring equipment such as power supplies, oscilloscope, etc. will be needed.

V. Profits: If the physical IC chip test results successfully meet the requirements, then it would move on to the market. The profits made from the chip would go to multiple parties. The people who designed and tested the circuit will receive a cut, but the profit will also be used to pay the manufacturing and delivery company.

If Manufactured on a Commercial Basis:
- Estimated Numbers of Devices Per Year: 100k
- Estimated Manufacturing Cost for Each Device: $3
- Estimated Purchase Price for Each Unit: $5
- Estimated Profit per Year: $200k
- Estimated Cost for User to Operate Devices: $0, very minimal electrical cost

Numbers are based on searching similar products like Sigma-Delta ADC on the current market. All numbers are assumed to be when the product is mass-produced with a contract with a fabrication company and delivery service. This will reduce the cost of manufacturing and increase the device sold per year.

Environmental:
It is important to note that IC chips are heavily reliant on rare materials like silicon, copper, gold, etc. The majority of these materials are only obtainable through mining which may affect the surrounding environment negatively. Also, the necessary chemicals needed to manufacture these die chips may be toxic and their waste may cause heavy environmental impact if not disposed of properly.

Manufacturability:
The manufacturability of IC chips doesn’t involve a lot of parties. It only requires a couple of different companies to get the whole process complete. One company will be needed to fabricate the IC chips from silicon die. Another company which may be the same as the one fabricating the IC chip will need to package all the individual silicon IC chips, so they can be mounted onto PCB boards. The last company needed is the delivery service for these chips. It will most likely be an electrical component-specific company such as Digi-Key since the product is not needed by the general public.

Sustainability:
When designing any product, it is important to keep sustainability as one of the most important requirements for the product. Not only does it satisfy the customers, but it will keep the product out of the landfill if the product has a longer life cycle with high-quality component are. Especially in ICs, it is important to have reliable and long-lasting products since they are essentially unfixable as soon as they break and not upgradable. Another important factor that
affects sustainability is the efficiency of the product. The higher the efficiency, the less power dissipated. With less power loss, it will require less input power in the first place for the product to operate. This will allow the saved power to be used in other places of the circuit. As real-life components are introduced to the circuit, it is important to test and observe how efficient these real-life components. If they are not satisfying the requirements, other methods or designs may need to be introduced.

Ethical:
The major ethical concern in this product is the environmental impact of the materials used to produce this CODEC. With the right methods and precautions, the CODEC will be able to be manufactured with minimal environmental impact, but it is impossible to have absolutely zero impact. Also, by taking into account the possible health and safety of the users beforehand, the manufacturers can implement safety circuits into the design to avoid any issues in this category.

Health and Safety:
The safety of the user is a must when selling a product. Since the design was only for simulation, no consideration is needed at this time. Once the design becomes manufacturable, some safety concerns may be fire hazards or explosions if the circuit is not wired properly or if there was a problem in the manufactured product. To avoid any fire hazards for user errors, it is important to place safety circuits such as over current protection or over-voltage protection within the design to avoid any components receiving anything over its ratings. To avoid any manufacturing errors, it’s important to make contracts with fabrications companies that are reliable and have strong quality inspections. Just because a company fabricates a chip relatively cheap, doesn’t mean their quality is just as good.

Social and Political:
There will be no political issues from this audio compression product, but it will still impact the current audio community and market. There are two major groups of stakeholders: the customers and the current audio hardware designers. The customers will be mostly audio hardware-related companies and if they find the product to be attractive and start implementing it into their products, then it will cause fewer sales for the current companies making digital signal compressing ICs.

Development:
Since the whole simulation was designed on Cadence Virtuoso, navigating around the environment because a breeze by the end of the project. Many tasks such as building a circuit, making an object symbol, editing simulation settings, etc. were all done on Cadence. Also, many components involved Verilog-A, so familiarity with the program had to be achieved. In the end, making new objects by initializing variables, using built-in functions, setting output voltage, etc. were done in Verilog-A.
Appendix B: Verilog-A Programming Found in Components

Comparator

This Code is based on the code found for comparator Cell found in ahdLib library from Cadence.

`include "discipline.h"
`include "constants.h"

// Based on the OVI Verilog-A Language Reference Manual, version 1.0 1996
//
// comparator
//
// - comparator
//
// sigin: input signal (positive terminal)
// sigref: reference signal (negative terminal)
// sigout: comparator output
//
// INSTANCE parameters
// sigout_high  = maximum output of the comparator
// sigout_low   = minimum output of the comparator
//
// observes the sigin with sigref.
// if sigin > sigref, it will output sigout_high value
// if sigout < sigref, it will output sigout_low value

module comparator_max_ver(sigin, sigref, sigout);

input sigin, sigref;
output sigout;
electrical sigin, sigref, sigout;
parameter real sigout_high = 10; // can change the output HIGH value
parameter real sigout_low = -10; // can change the output LOW value

analog begin
@ ( initial_step ) begin
// makes sure that the setted output HIGH is lower than the output LOW
if (sigout_high <= sigout_low) begin
  $display("Range specification error. sigout_high = (%E) less than sigout_low = (%E).\n", sigout_high, sigout_low );
  $finish;
end
end

// if input signal is higher than the reference signal
if (V(sigin) > V(sigref)) begin
  V(sigout) <+ sigout_high; // output HIGH
end
else // everything else
  V(sigout) <+ sigout_low; // output LOW
end
endmodule
ClockCounter

// VerilogA for EE461Project, ClockCounter, veriloga

`include "constants.vams"
`include "disciplines.vams"

// CLK: the input CLK signal
// Inter: the output interrupt signal
//Counts the rising edges of the CLK signal and will output a
//brief HIGH signal from Inter pin once it counts 64 instances
//of the rising edge

module ClockCounter(CLK, Inter);
input CLK;
output Inter;
electrical CLK, Inter;

//Transition Parameters
parameter real Td = 0; //input-to-output time delay
parameter real Tr=1n from (0:inf); // output rise time
parameter real Tf=Tr; // output fall time
parameter real ttol=Tr/10; // edge detect time tolerance
parameter real vtol=0.1; // edge detect voltage tolerance

//Input Parameters
parameter real Vhigh = 5; //sets what the HIGH is for CLK
parameter real Vlow = 0; //sets what the LOW is for CLK
//calculates the center of HIGH and LOW
parameter real Vth = 0.5*(Vhigh+Vlow);
//how many CLK cycles for Inter to go HIGH
parameter integer dec_ratio = 64;

integer CLK_counter; //used to store the CLK HIGHS

analog begin
//sets count to zero
@(initial_step("static","tran")) begin
  CLK_counter = 0;
end
//for every rising edge, adds one to count
@(cross(V(CLK)-Vth,1,ttol,vtol)) begin
  CLK_counter = CLK_counter + 1;
end
//compares if count is equal to dec_ratio
//if equal, will set Inter as HIGH
V(Inter) <+ transition(CLK_counter&(dec_ratio)? Vhigh:Vlow, Td,Tr,Tf);
//resets count to be zero after HIGH inter
if (CLK_counter == dec_ratio) begin
  CLK_counter = 0;
end
endmodule

SimpleAveraging

// VerilogA for EE461Project, CIC_Filter, veriloga
'include "constants.vams"
#include "disciplines.vams"

//data_in: input data-stream
//CLK_int: interrupt signal
//data_outb0: bit 0 output
//data_outb1: bit 1 output
//data_outb2: bit 2 output
//data_outb3: bit 3 output
//data_outb4: bit 4 output
//data_outb5: bit 5 output
//onecountertest: testing pin, output decimal form value

module SimpleAveraging (data_in, CLK_int,data_outb0, data_outb1, data_outb2, data_outb3, data_outb4, data_outb5,onecountertest);
    input data_in, CLK_int;
    output data_outb0, data_outb1, data_outb2, data_outb3, data_outb4, data_outb5,onecountertest;
electrical data_in, CLK_int, data_outb0, data_outb1, data_outb2, data_outb3, data_outb4, data_outb5, onecountertest;

//Transition Parameters
parameter real Td = 0;       //input-to-output time delay
parameter real Tr=1n from (0:inf); // output rise time
parameter real Tf=Tr;        // output fall time
parameter real ttol=Tr/10;   // edge detect time tolerance
parameter real vtol=0.1;     // edge detect voltage tolerance

//Input Parameters
parameter real Vhigh= 2.5;   //sets what the HIGH is for input
parameter real Vlow = -2.5;   //sets what the LOW is for input
parameter real Vth =0.5*(Vhigh+Vlow); //logic threshold levels
integer one_counter;        //counts HIGH from data_in
integer totalcounts;        //temporarily transfers total count for outputting

analog begin
    //sets count to zero
    @(initial_step("static","tran")) begin
        one_counter = 0;
    end
    //when rising edge is detected from data_in, add one to count
    @(cross(V(data_in) - Vth,1,ttol,vtol)) begin
        if (V(data_in) > Vth) begin
            one_counter = one_counter + 1;
        end
    end
end
// when interrupt is detected
if (V(CLK_int) > Vth) begin
    // subtracts one to account for zero
    totalcounts = one_counter - 1;
    // if zero, set totalcounts to zero
    if (one_counter == 0) begin
        totalcounts = 0;
    end
    // resets count to zero
    one_counter = 0;
end
// outputs the bit's HIGHs and LOWs depending on totalcount
V(data_outb0) <+ transition(totalcounts & 1? Vhigh:Vlow, Td, Tr, Tf);
V(data_outb1) <+ transition(totalcounts & 2? Vhigh:Vlow, Td, Tr, Tf);
V(data_outb2) <+ transition(totalcounts & 4? Vhigh:Vlow, Td, Tr, Tf);
V(data_outb3) <+ transition(totalcounts & 8? Vhigh:Vlow, Td, Tr, Tf);
V(data_outb4) <+ transition(totalcounts & 16? Vhigh:Vlow, Td, Tr, Tf);
V(data_outb5) <+ transition(totalcounts & 32? Vhigh:Vlow, Td, Tr, Tf);
V(onecountertest) <+ transition(one_counter, Td, Tr, Tf);
end