Design of a 4-bit Digital to Analog Converter

Zoe Jaramillo
Electrical Engineering
California Polytechnic State University
San Luis Obispo, CA USA
zojarami@calpoly.edu

Robert Tolmanov
Electrical Engineering
California Polytechnic State University
San Luis Obispo, CA USA
rtolmano@calpoly.edu

***Abstract*— Digital to Analog Converters (DACs) convert binary inputs into analog signals, allowing digital devices to control real-world systems. DACs can be used in audio and video applications. For our project, we used a Weighted Resistor DAC, which maps binary inputs to voltages using scaled resistors. The design is simple, affordable, and easy to understand, making it ideal for learning.**

Keywords—DAC

I. INTRODUCTION

Digital-to-Analog Converters (DACs) are circuits that convert a binary input into an analog output. This is useful since many real-world systems function in analog form, allowing digital devices like microcontrollers or computers to effectively interact with and control them. DACs are widely used in applications like audio playback, video displays, and communication systems, where accurate analog output is crucial for functionality [4].

There are two main types of DACs: the Weighted Resistor DAC and the R-2R Ladder DAC. The difference between the two is how they generate the analog output. Weighted Resistor DACs use resistors scaled by powers of two, while R-2R Ladder DACs use only two resistor values in a repeating structure [5]. Our design used the Weighted Resistor DAC, which made it straightforward to map binary inputs to proportional voltages and easier to understand and verify the circuit's behavior.

This DAC design stands out for its simplicity, low cost, and educational value. By using readily available components and a straightforward layout, it's easy to assemble and understand, making it perfect for students learning about digital-to-analog conversion. Its compact size and low power consumption also make it a great fit for compact systems and small-scale prototypes.

Figure 2: DAC Simulation Output

Our paper is structured to guide the reader through the step-by-step process of our DAC design and implementation. Figure 1 and Figure 2 show the LTspice schematic and simulated output, which we used as a starting point for the actual circuit. Using this test model, we created a schematic in EAGLE (Fusion 360) and converted it into a PCB layout for our physical design.

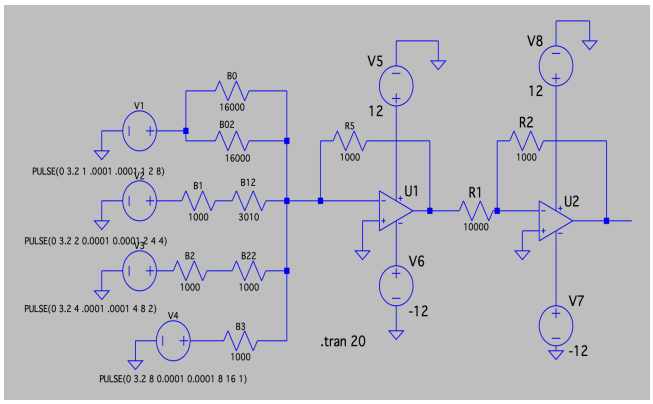

Figure 1: 4-bit DAC Schematic

II. 4-BIT DAC DESIGN AND SIMULATION

Figure 1 *s*hows our LTspice schematic [6], where we used a 1000 Ω feedback resistor, matching the resistor connected to the most significant bit (Bit3). The other bits were scaled accordingly: Bit2 used 2000 Ω, Bit1 used 4000 Ω, and Bit0 used 8000 Ω. Only the 4000 Ω resistor slightly deviated from the ideal, measuring 4010 Ω, which was still within the ±5% tolerance range. This resistor arrangement allowed the DAC to function correctly, as confirmed by the LTspice simulation, which produced the expected analog output for binary inputs ranging from 0 to 15 (Figure *2).*

Table 3 displays the LTspice simulation results alongside the calculated theoretical output voltages for all 16 possible 4-bit binary inputs. The simulated outputs closely match the theoretical values, with a maximum deviation of only ±0.003V.

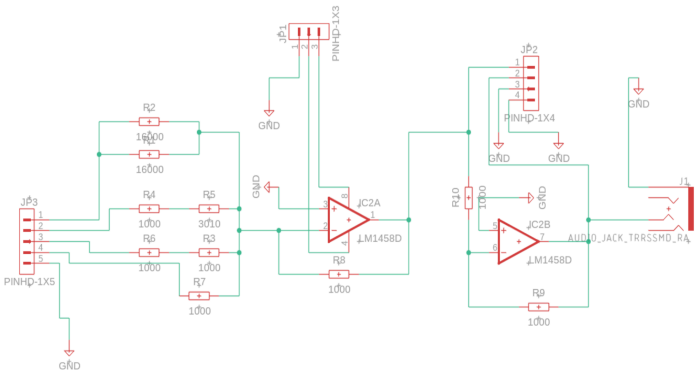

Figure 4: 4-bit DAC Eagle Schematic

III. PCB DESIGN

We used EAGLE PCB Design [1] to design the PCB for the 4-bit DAC circuit. The EAGLE 4-bit DAC schematic is shown in Figure 4. To be able to fit the whole design in a 1" x 1" square, we selected to use surface mount 1206 size components for all resistors. We also selected surface mount components for the ICs.

To power the circuit, we created a three-pin header for +/- 12V and ground. We created a four-pin header for the outputs of the two op-amps and the two grounds. We also created a five-pin header for the four DAC inputs and ground.

From the schematic, we generated a 2-layer PCB board layout as shown in Figure 5. After manually placing the components in a given location, we used EAGLE's auto-router with EAGLE's default design rules file (which governs the trace width, the drill size, and the clearance between traces and components) to route the design.

We added ground pours on both layers to improve heat dissipation and placed additional vias around the board (shown as the unconnected green dots in Figure 5) to ensure a solid connection between the ground pours on both layers. The board measures 1" x 1", meeting the size limitation. We placed the components in this specific order to minimize the number of vias. Fewer vias improve signal flow and make the board easier to fabricate. In our case, we ended up having 8 vias. After routing the design, we used EAGLE's Design Rule Check (DRC) tool to verify that the layout met all required design rules.

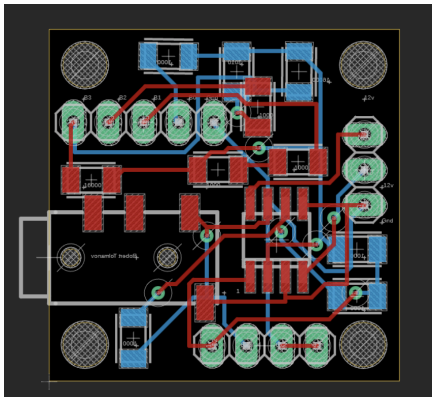

Figure 5: 4-Bit DAC Eagle Board Layout

Input (Bit 3,2,1,0)	Theoretical Output Voltage (V)	Simulated Output Voltage (V)	Hardware Output Voltage (V)
0000	0.0	0.000	0.000
0001	0.4	0.399	.450
0010	0.8	0.798	.900
0011	1.2	1.197	1.35
0100	1.6	1.600	1.75
0101	2.0	1.999	2.1
0110	2.4	2.398	2.6
0111	2.8	2.797	3.00
1000	3.2	3.201	3.5
1001	3.6	3.599	3.8
1010	4.0	3.998	4.3
1011	4.4	4.397	4.7
1100	4.8	4.800	5.25
1101	5.2	5.2	5.475
1110	5.6	5.6	5.7
1111	6.0	6.0	6.3

Table 3: Simulated and Theoretical Values of the Op-Amp Outputs Using a Binary Counter as the Input

IV. PCB ASSEMBLY AND TESTING

We ordered all of the electronic components from Digikey and ordered the PCBs from OshPark[3] – a low-cost PCB fabrication company. OshPark charges $5 per square inch for two-layer boards and gives you three boards for each purchase. Therefore, the total cost for the circuit was $17.

To assemble the PCBs, we used a soldering iron with leaded solder to mount the surface components. Solder paste was applied to all components except the headphone jack, which was the only part soldered manually using a hand-soldering technique to ensure a strong mechanical connection. This approach made the assembly efficient while keeping the build secure and functional. Figure 7 shows the assembled PCB.

We powered the board with a ±12V supply and checked for proper operation by connecting the DAC to a binary counter. This produced the expected staircase waveform at both op-amp outputs on the oscilloscope, closely matching the behavior seen in our LTspice simulation (Figures 2 and 6).

Next, we tested the system by sending a sinusoidal input to one of the Arduino's analog pins. The Arduino reads the signal and outputs it to the logic analyzer channels, shown in Figure 8. For the final setup, the level-shifting circuit was wired to the Arduino's analog input, which then passed the

signal to the DAC. The resulting output was successfully heard through the speaker, confirming proper functionality (Figure 10).

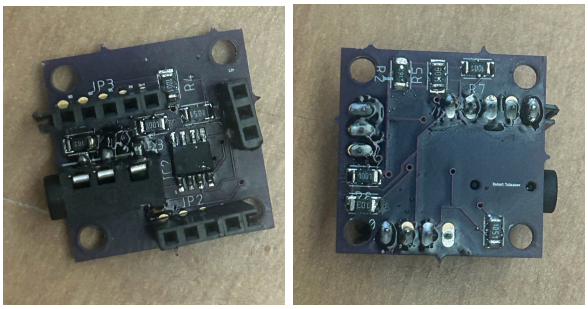

Figure 7: Fully Assembled OshPark PCB

V. 4-BIT DAC APPLICATION

We used the DAC to convert analog signals into digital ones. Our DAC is taking values from the Arduino's analog converters and turning them into 16 levels to make a 4-bit digital version of the input analog signal. Figure 12 shows an input Sine Wave and its digitized outputs of the DAC below it. We would later run an audio signal through our DAC board (Figure 10) and send the output to a speaker, where we could hear a 4-bit digital version of the analog audio signal.

You can also see the output of a square wave (Figure 8) run through our DAC. The top orange signal is the input wave, and the graphs below represent the DAC's output after processing that wave. You can see the output is similar to the input since a square wave acts similarly to a digital waveform.

We can also see in Figure 13 a full digitized Sine Wave, running at 1 kHz, 5Vpp with 2.6V offset. These various tests show our DAC board digitizes and outputs an analog signal. Making these signals digital is very useful for anyone working with audio files, as digital signals are easier to manipulate and edit as opposed to analog ones. This DAC is a simple version of how music is edited across various industries.

Figure 9 shows the block diagram of the complete audio system after all components were connected. If we changed our system to have a 12-bit analog-to-digital converter and a 6-bit digital-to-analog converter, we would use the Arduino code shown in Figure 11.

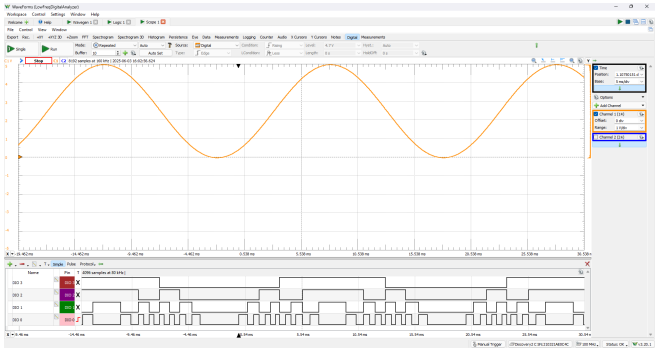

Figure 12: A Sine wave and the represented 4-bit digital form

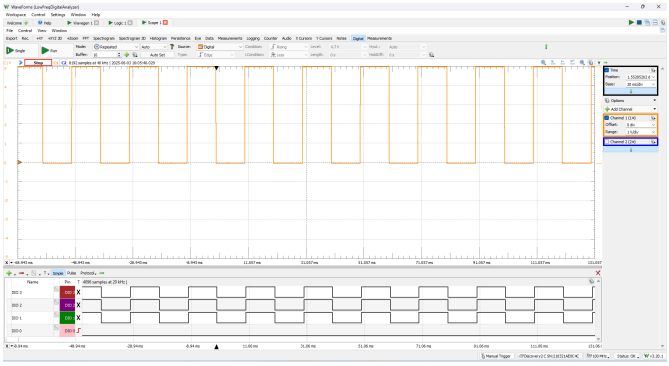

Figure 8: A Square Wave and the represented 4-bit digital form

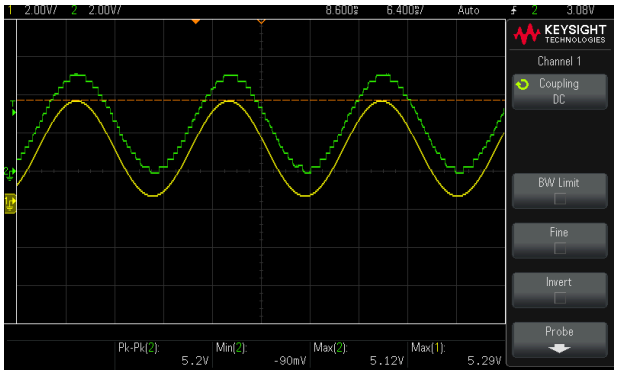

Figure 13: Oscilloscope output of a 1 kHz sine wave after DAC, showing a 4-bit digitized signal

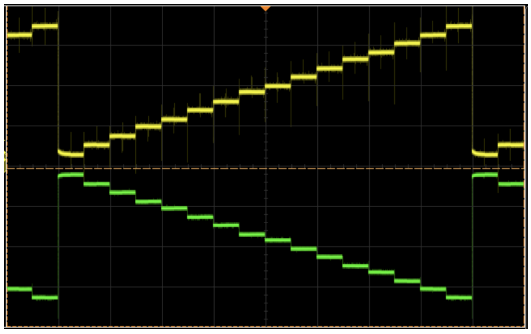

Figure 6: Oscilloscope output showing all 16 DAC voltage levels for 4-bit input values

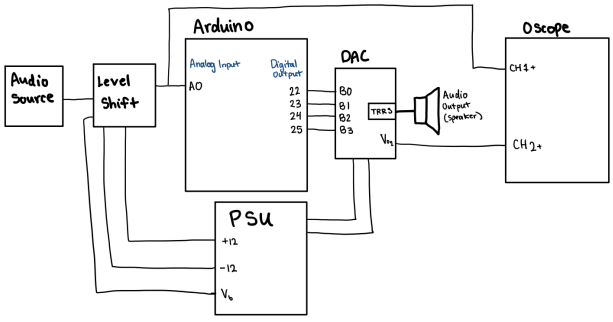

Figure 9: Block Diagram of the Complete Audio System

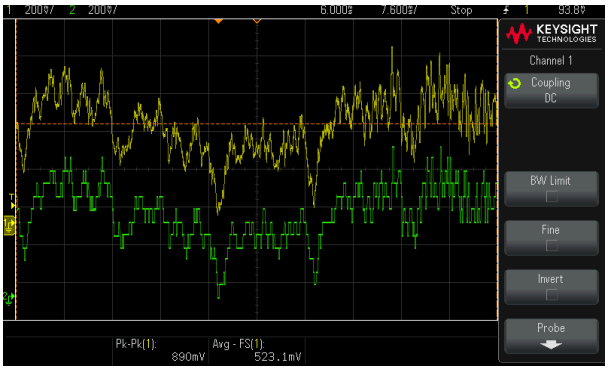

Figure 10: DAC Output and Input signal of Audio file, 1V peak-to-peak, 500 mV average (fs).

VI. SUSTAINABILITY

The DAC audio system is sustainable because it uses low-power, affordable parts and open-source tools, making it accessible and easy to reuse. It follows green engineering principles like reducing energy use, minimizing waste, and designing with simplicity in mind. The project also considers safety and educational value, making it a practical and eco-friendly solution for learning environments.

VII. CONCLUSION

This project demonstrated how a simple, low-cost weighted resistor DAC can convert digital signals into analog outputs, making it an effective way to explore real-world uses of digital-to-analog conversion. By building and testing the system with audio signals, we gained hands-on experience in signal processing and circuit integration. The project was useful for reinforcing key concepts in electronics and digital signal processing through hands-on application. In the future, the system could be improved by using higher-resolution DACs or adding filtering to smooth the output signal for better audio quality.

```
// the setup routine runs once when you press reset:
void setup() {
  // Set pins 22 to 27 (lower 6 bits of Port A) to output
  DDRA = 0x3F; // 0b00111111
  // Set reference voltage to match the scale of the input signal
  analogReference(DEFAULT);
}

// the loop routine runs over and over again forever:
void loop() {
  // Read the 12-bit input on analog pin 0
  int sensorValue = analogRead(A0);
  // Convert the 12-bit value (0-4095) to a 6-bit value (0-63)
  int output = (int)(sensorValue * 0.01538); // 63 / 4095
  // Output the 6-bit value to the DAC
  PORTA = output & 0x3F; // To keep only lower 6 bits
}
```

Figure 11: Arduino code for 12-bit ADC input to 6-bit DAC Output

ACKNOWLEDGMENT

The authors would like to thank Jeremy Blum for creating excellent YouTube tutorials for EAGLE [1]. I would also like to thank the Cal Poly Electrical Engineering department for procuring useful lab manuals and resources.

REFERENCE

[1] EAGLE PCB Design. https://www.autodesk.com/
[2] SparkFun Electronics. https://www.sparkfun.com/
[3] Osh Park – An electric ecosystem. https://oshpark.com/
[4] Electricity and Magnetism https://www.electricity-magnetism.org/digital-to-analog-converters-dac/
[5] Tutorials Point https://www.tutorialspoint.com/linear_integrated_circuits_applications/linear_integrated_circuits_applications_digital_to_analog_converters.htm
[6] LTspice https://www.analog.com/en/index.html