深入理解并实现基本的模数转换器（ADC）原理与实现

黄京

Oct 14, 2025

探索采样、量化与编码的奥秘，并用电路和代码亲手实现一个简单的 ADC。
我们生活在一个充满连续变化的模拟世界中，温度的高低、声音的强弱、光线的明暗都以模拟形式存在。然而，计算机和数字系统只能处理离散的二进制数据，即 0 和 1。模数转换器（ADC）正是连接这两个世界的桥梁，它负责将连续的模拟信号转换为离散的数字信号，使得物理世界的信息能够被数字设备理解和处理。ADC 在现代技术中无处不在，例如智能手机的触摸屏通过 ADC 检测触摸位置，麦克风录音时将声波转换为数字音频流，数字体温计测量体温并显示数值，物联网传感器采集环境数据并上传到云端。这些应用都依赖于 ADC 的精确转换能力。本文旨在深入讲解 ADC 的核心工作原理，包括采样、量化和编码三个关键步骤，并介绍几种主流 ADC 类型及其优缺点。最终，我们将引导读者通过软件模拟和硬件理解的方式，实现一个基本的逐次逼近型 ADC 模型，从而将理论知识转化为实践技能。

1　ADC 的核心三部曲：采样、量化、编码

模数转换过程可以概括为三个基本步骤：采样、量化和编码。这些步骤共同作用，将连续的模拟信号转换为离散的数字代码。首先，采样阶段负责在时间维度上捕捉模拟信号的瞬时值。采样以固定的时间间隔进行，这个间隔的倒数称为采样频率。采样频率的选择至关重要，它必须遵循奈奎斯特-香农采样定理，即采样频率至少是信号最高频率的两倍，以避免混叠现象。混叠会导致高频信号被错误地表示为低频信号，从而失真。例如，如果一个模拟信号的最高频率成分是 1000 Hz，那么采样频率至少应为 2000 Hz 才能准确重建信号。在实际应用中，如果采样频率过低，原本的高频正弦波可能会被误判为低频波形，造成数据错误。
接下来是量化阶段，它将采样得到的连续电压值映射到有限个离散的电压等级中。量化过程引入了分辨率的概念，分辨率通常用位数表示，例如一个 8 位 ADC 可以将电压范围划分为 (2^8 = 256) 个等级。假设一个 3 位 ADC 的参考电压范围为 0 到 5 V，那么它会将这个范围分成 8 个等间隔的区间，每个区间对应一个离散等级。例如，0 到 0.625 V 对应等级 0，0.625 到 1.25 V 对应等级 1，依此类推，直到 4.375 到 5 V 对应等级 7。量化过程中不可避免会产生量化误差，这是因为连续的电压值被“四舍五入”到最接近的离散等级。量化误差的最大值为 ±1/2 LSB（最低有效位），例如在上述 3 位 ADC 中，每个等级的步长为 0.625 V，因此最大误差为 ±0.3125 V。这种误差是 ADC 固有的，无法完全消除，但可以通过提高分辨率来减小。
最后，编码阶段将量化后的等级值转换为二进制代码，以便数字系统处理。继续以 3 位 ADC 为例，量化等级 0 可能编码为二进制 000，等级 1 为 001，一直到等级 7 为 111。编码过程通常使用标准二进制或补码格式，具体取决于应用需求。通过这三个步骤，ADC 成功地将模拟信号转换为数字形式，为后续的数字处理和分析奠定了基础。

2 常见 ADC 类型巡礼

在模数转换领域，有多种 ADC 架构可供选择，每种都有其独特的优缺点和适用场景。Flash ADC 是一种高速转换器，它使用一列比较器并行比较输入电压与参考电压，从而实现极快的转换速度。然而，Flash ADC 的电路复杂度随分辨率指数增长，例如一个 8 位 Flash ADC 需要 255 个比较器，导致高功耗和高成本，因此它主要用于超高速应用如示波器和雷达系统。

逐次逼近型 ADC（SAR ADC）在速度、精度和成本之间取得了良好平衡，因此成为最通用的 ADC 类型之一。它的工作原理类似于天秤称重，从最高位开始逐位试探和比较输入电压。SAR ADC 通过一个数模转换器（DAC）生成试探电压，并与输入电压比较，根据比较结果调整二进制代码。这种架构速度适中，广泛应用于微控制器和数据采集系统。本文将重点介绍并实现这种 ADC 类型。

双积分型 ADC 采用电压-时间转换原理，通过两次积分过程将输入电压转换为时间间隔，再通过计时得到数字值。这种 ADC 具有高精度和强抗干扰能力，但转换速度较慢，因此适用于数字万用表和高精度测量仪器。此外，Sigma-Delta ADC 使用过采样和噪声整形技术，以高速 1 位 ADC 为基础实现高分辨率。它成本较低，但建立时间慢，常用于音频采集和高精度传感器应用。

3 动手实践：实现一个简易的逐次逼近型 ADC

逐次逼近型 ADC 的核心思想是二分搜索算法，它通过逐位比较来逼近输入电压值。首先，ADC 从最高位开始，将该位置 1，其余位为 0，生成一个试探电压。然后，通过 DAC 将试探值转换为模拟电压，并与输入电压比较。如果试探电压小于或等于输入电压，则保留该位为 1；否则，清除该位为 0。接着，移动到下一位，重复这个过程，直到所有位都被处理完毕。最终，得到的二进制代码就是输入电压的数字表示。例如，在一个 4 位 ADC 中，如果输入电压为 2.7 V，参考电压为 5 V，那么转换过程可能从二进制 1000（对应 2.5 V）开始，逐步调整到 1010（对应 3.125 V），最后得到 1001（对应 2.8125 V），完成逼近。

现在，我们通过 Python 代码模拟 SAR ADC 的逻辑。以下代码定义了一个简单的 SAR ADC 函数，它接受输入电压、参考电压和分辨率作为参数，并返回数字输出。代码首先初始化数字输出和当前位，然后通过循环逐位进行试探和比较。在每次迭代中，代码生成试探值，将其转换为模拟电压（通过简单的计算模拟 DAC），并与输入电压比较，根据结果更新数字输出。最后，函数返回最终的二进制代码。

```
def sar_adc(input_voltage, vref, bits):
    digital_output = 0
    current_bit = bits - 1 # 从最高位开始
    while current_bit >= 0:
        # 将当前位置 1，生成试探值
        trial = digital_output | (1 << current_bit)
        # 将试探值转换为模拟电压（模拟 DAC 输出）
        trial_voltage = (trial / (2**bits)) * vref
        # 与输入电压比较
        if trial_voltage <= input_voltage:
            digital_output = trial # 保留该位
        current_bit -= 1 # 移至下一位
```

```
    return digital_output

# 示例使用
vref = 5.0 # 参考电压 5V
bits = 4 # 4 位分辨率
input_voltage = 2.7 # 输入电压 2.7V
result = sar_adc(input_voltage, vref, bits)
print(f"输入电压 {input_voltage} V 的数字输出为 : {bin(result)}")
```

在这段代码中，我们首先定义函数 sar_adc，它使用 digital_output 变量存储当前数字值，current_bit 表示当前处理的位位置。循环从最高位（bits-1）开始，到最低位（0）结束。在每次循环中，我们使用位操作 (1 << current_bit) 将当前位设为 1，然后通过 trial_voltage = (trial / (2**bits)) * vref 计算试探电压，这模拟了 DAC 的转换过程。接着，比较试探电压与输入电压，如果试探电压小于或等于输入电压，则更新 digital_output 以保留该位。最后，函数返回数字输出。运行示例后，对于输入电压 2.7 V，代码可能输出二进制 0b1001，表示数字值 9，对应电压约 2.8125 V，体现了量化误差。

在硬件层面，SAR ADC 由逐次逼近寄存器、数模转换器、比较器和控制逻辑组成。工作时，时钟信号驱动控制逻辑，逐位生成试探代码，DAC 将其转换为模拟电压，比较器判断大小，并根据结果更新寄存器。这种电路结构在微控制器中常见，例如 Arduino 的 analogRead() 函数内部就使用了类似的 ADC。通过在线仿真工具如 Falstad Circuit Simulator，可以搭建并观察 SAR ADC 的时序行为，加深理解。

4 ADC 的关键性能参数解读

ADC 的性能由多个参数描述，这些参数直接影响转换精度和速度。分辨率是 ADC 能够区分的最小电压变化，通常用位数表示，例如一个 10 位 ADC 在 0 到 5 V 范围内的分辨率约为 4.88 mV。分辨率越高，量化误差越小，但转换可能更慢或成本更高。采样率指每秒采样的次数，它必须满足奈奎斯特定理以避免混叠，例如在音频应用中，采样率通常设为 44.1 kHz 以覆盖人耳可闻频率范围。

信噪比（SNR）衡量 ADC 输出信号与噪声的比率，它与分辨率密切相关，近似公式为 (SNR \approx 6.02N + 1.76) dB，其中 N 是 ADC 的位数。例如，一个 12 位 ADC 的理想 SNR 约为 74 dB，表示信号强度远高于噪声。有效位数（ENOB）则反映 ADC 在实际应用中的真实性能，它可能低于标称位数，因为实际电路会引入噪声和非线性误差。理解这些参数有助于在选择 ADC 时权衡速度、精度和成本。

本文从模拟世界与数字世界的连接需求出发，详细讲解了 ADC 的核心原理，包括采样、量化和编码三个步骤，并介绍了多种 ADC 类型，重点实现了逐次逼近型 ADC 的软件模拟和硬件理解。ADC 作为物理世界与数字系统交互的关键组件，其重要性不言而喻，它使得温度、声音和光线等模拟量能够被计算机处理和分析。展望未来，读者可以进一步在真实硬件上实践，例如使用 Arduino 的 analogRead() 函数测量光敏电阻或电位器，将理论知识应用于实际项目。通过不断探索和实验，我们能够更深入地理解 ADC 在现代技术中的核心作用，并推动创新应用的发展。如果您有任何问题或项目经验，欢迎分享和讨论。