**Analogue to Digital and Digital to Analogue Converters and Circuits**

Table of Contents

[Analogue-to-Digital Converters 3](#_Toc100173032)

[Accuracy 4](#_Toc100173033)

[Flash ADC 5](#_Toc100173034)

[Delta-Sigma ADC 7](#_Toc100173035)

[Successive Approximation ADC 8](#_Toc100173036)

[Digital-to-Analogue Converters 11](#_Toc100173037)

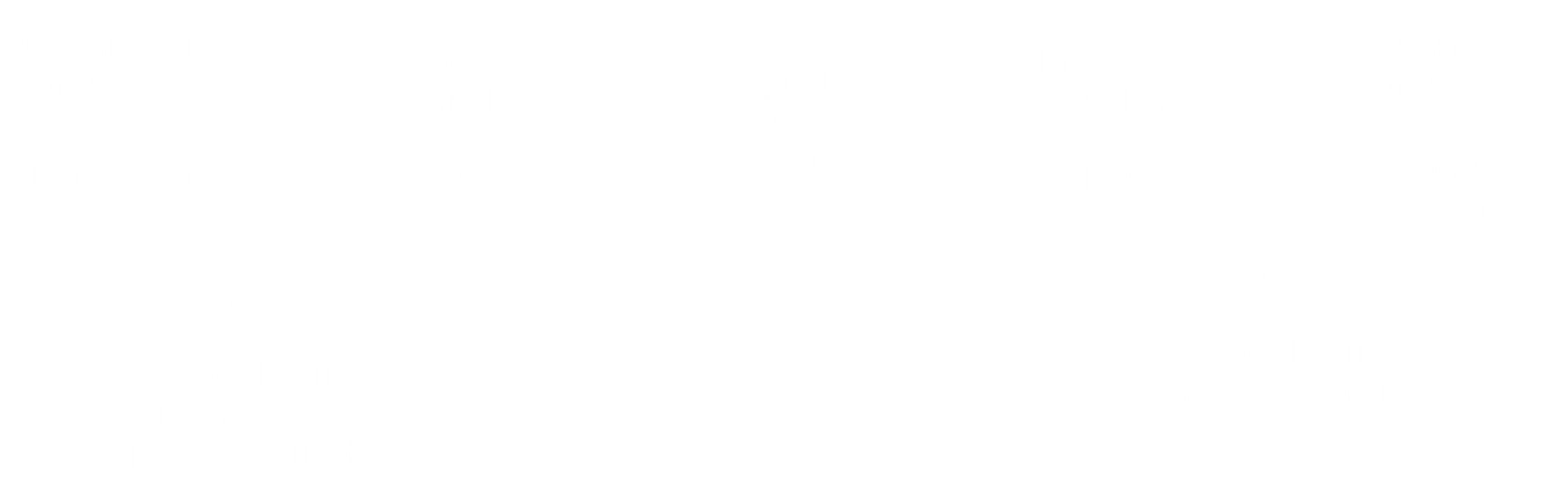
[Weighted Summing Amplifier DAC 13](#_Toc100173038)

[R-2R Ladder DAC 14](#_Toc100173039)

There are two types of signals, **Analogue**, which have a continuous range of values, and **Digital**, which have discrete values.

Analogue signals in particular are generally measured with respect to some other value. For example, the temperature of a room may be measured using a mercury thermometer using the value of the height of mercury.

Most data in the world is analogue. In order to use this data, we must convert it to digital data on the input end and back to analogue on the output end.



An **Analogue-to-Digital Converter** (ADC) converts analogue signals to digital code while a **Digital-to-Analogue Converter** (DAC) converts digital code to analogue signals.

Both ADCs and DACs can be connected to microprocessors using **interfaces**.

## Analogue-to-Digital Converters

When converting an analogue signal into a digital one, there are two steps, quantization and encoding.

In **quantization**, we divide the range of analogue values into sets, called **states**.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Output**  **States** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **Discrete Voltage**  **Ranges (V)** | 0.00  -  1.25 | 1.25  -  2.50 | 2.50  -  3.75 | 3.75  -  5.00 | 5.00  -  6.25 | 6.25  -  7.50 | 7.50  -  8.75 | 8.75  -  10.00 |

The number of states, , must be a multiple of , i.e. , where is the number of bits we will use in the encoding stage.

In the **encoding** stage, we measure the value of the analogue signal at regular time intervals and see which state the value falls into. Depending on the state, we assign a binary value.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Output**  **States** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **Output Binary**  **Equivalent** | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |

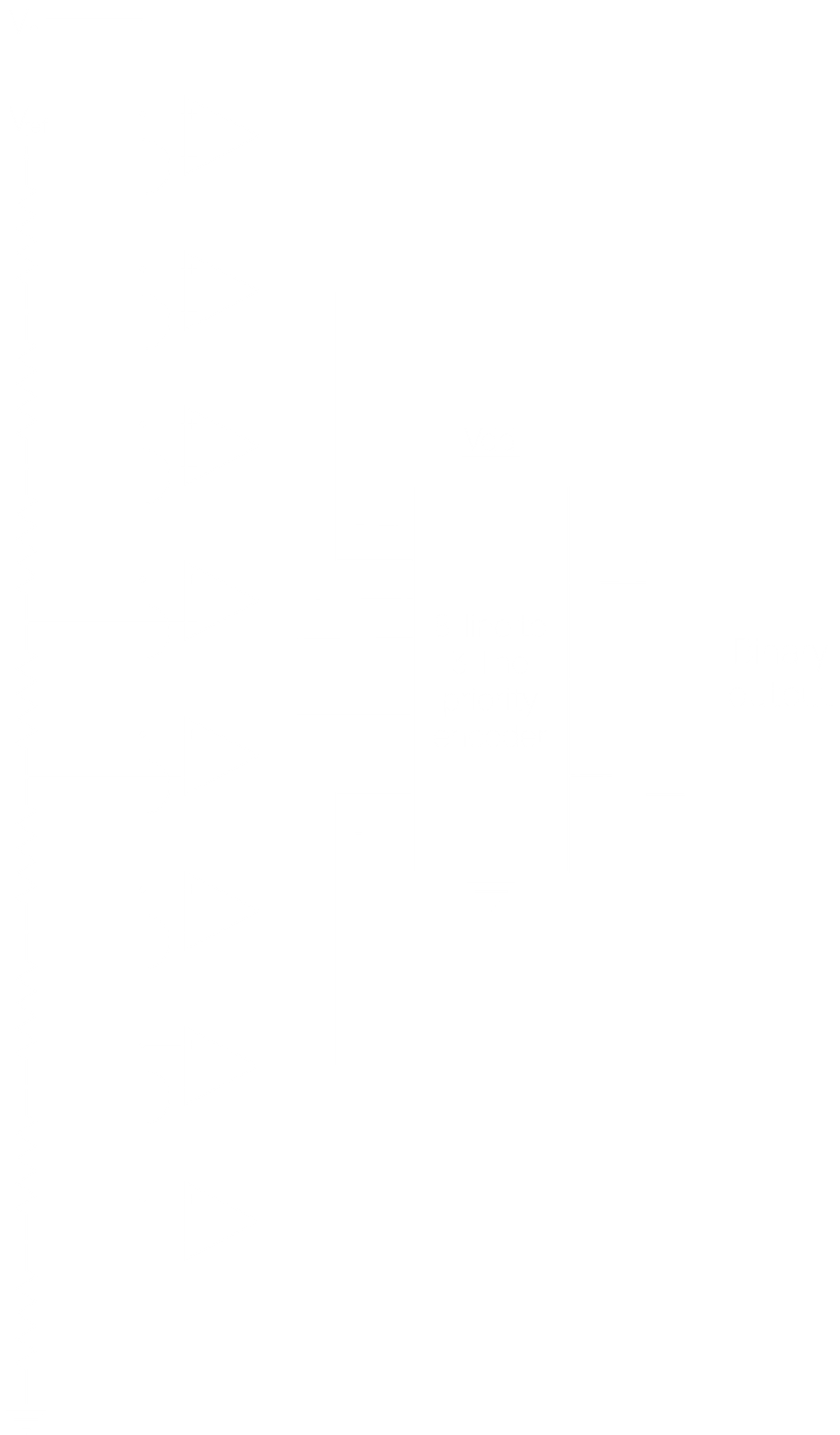
### Accuracy

There are two ways in which we can ensure that the digital signal more accurately represents the analogue one.

1. Increasing **resolution** – If we use more bits to represent the digital signal, the ranges of values in each state will be smaller. This will result in us being able to represent smaller changes in values.
2. Increasing **sampling rate** – If we take samples more often, it is more likely that we will catch a quick change in the analogue signal.

The latter improvement also ties in with fixing the **aliasing** problem. Aliasing occurs when the analogue signal is changing much faster than the rate at which we are sampling it. The **Nyquist Rule** states that the sampling rate should be at least twice the maximum frequency of the analogue signal to avoid aliasing.

## Flash ADC

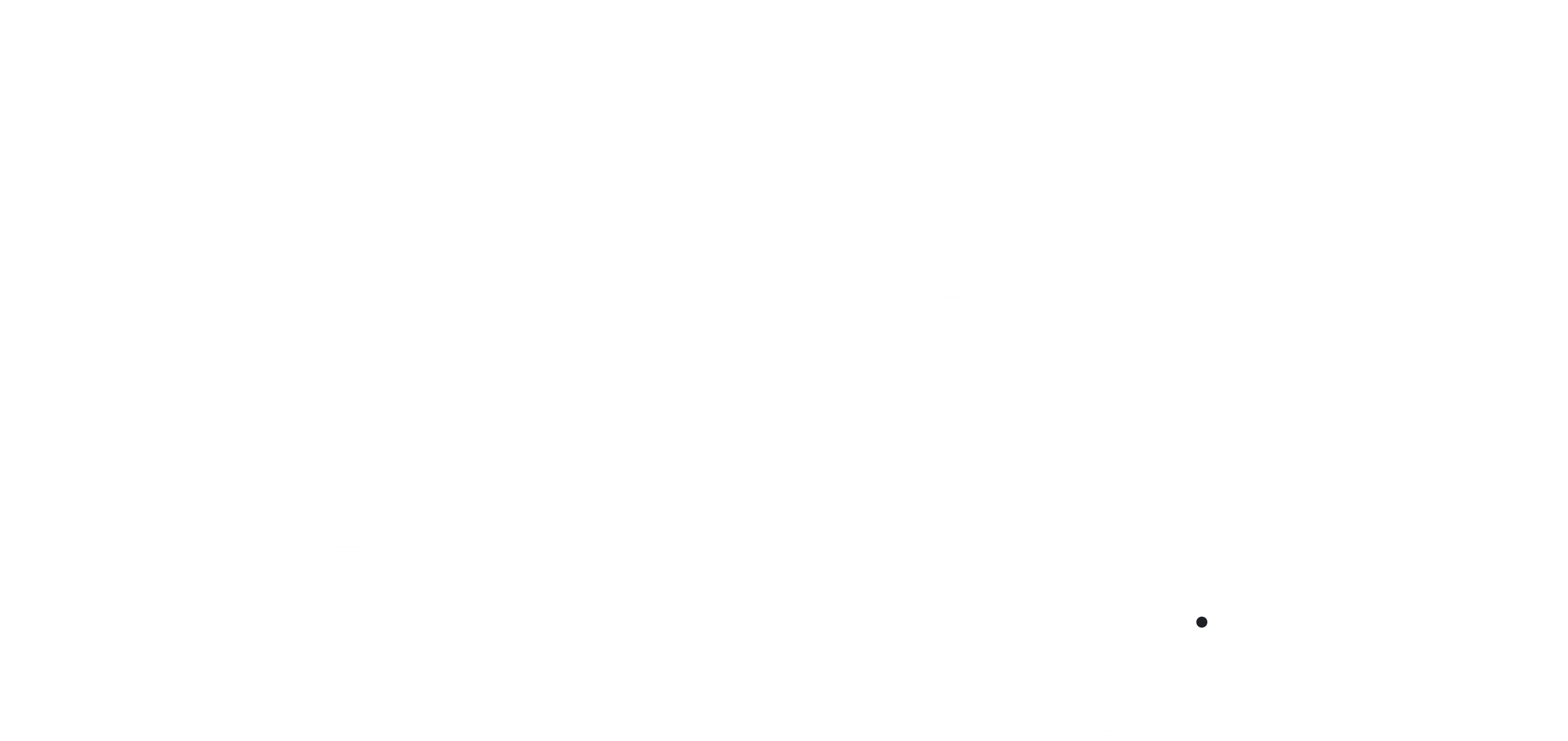


The **Flash ADC** has been given its name because it is extremely fast. There is a series of **comparators**, each comparing the voltage to a specific reference voltage. The input voltage is compared at each comparator in turn, which results in high outputs from the comparators from the point where the input voltage begins to exceed the reference voltage. Based on this, a **priority encoder** generates a binary output.

Even though Flash ADCs are fast, they require comparators, one for each possible state. This makes them infeasible for more than 5 or 6 bit outputs.

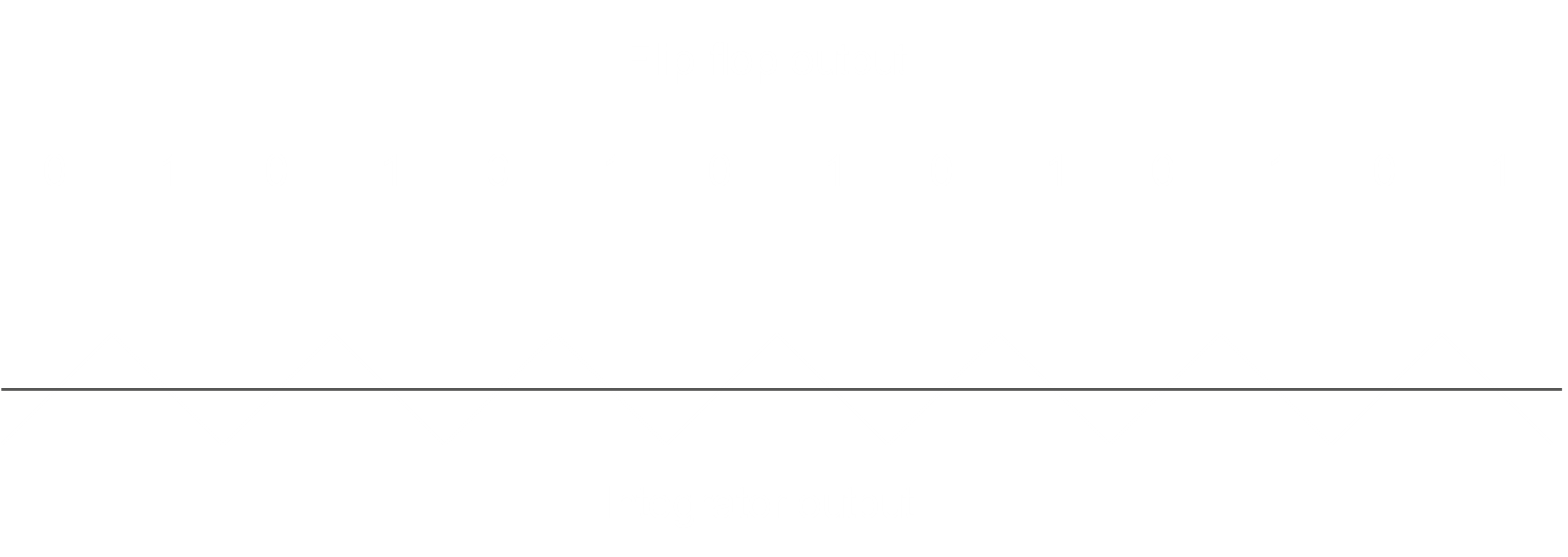
* Simplest in terms of operational theory
* Most efficient in terms of speed
* Lower resolution
* Expensive
* Limited in terms of comparator and gate propagation delays, i.e. output will be delayed for larger number of bits since signal must travel down longer path
* For each additional output bit, the number of comparators is doubled

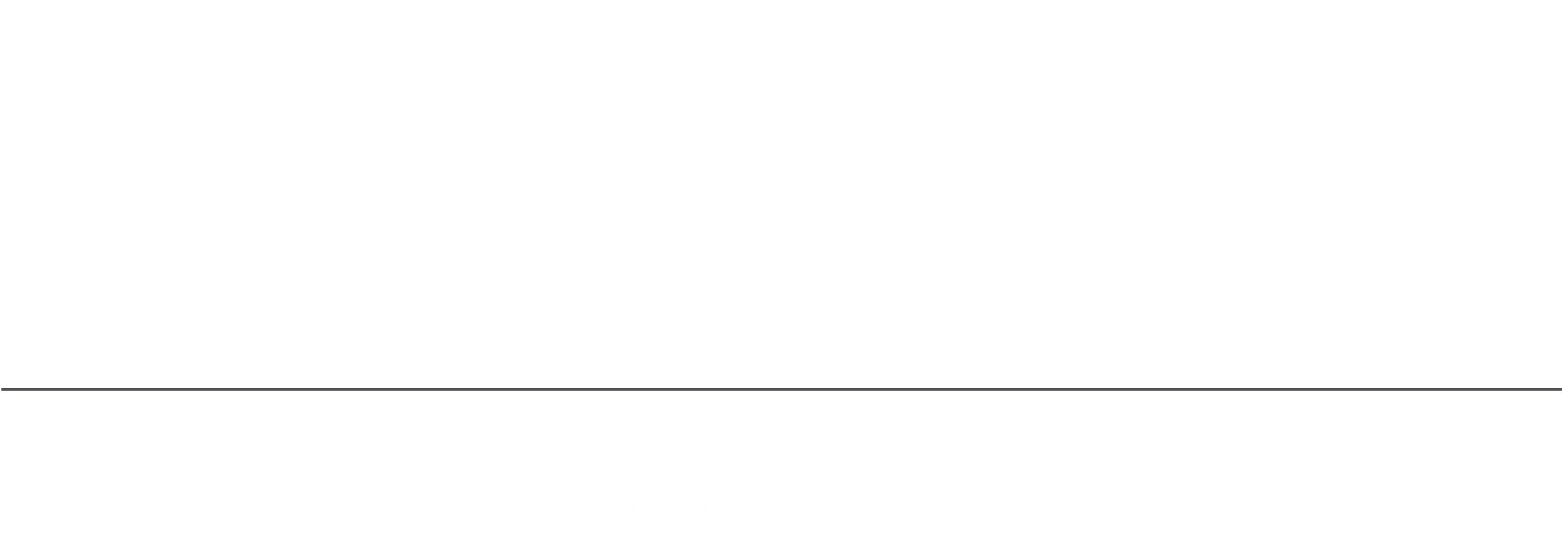
## Delta-Sigma ADC

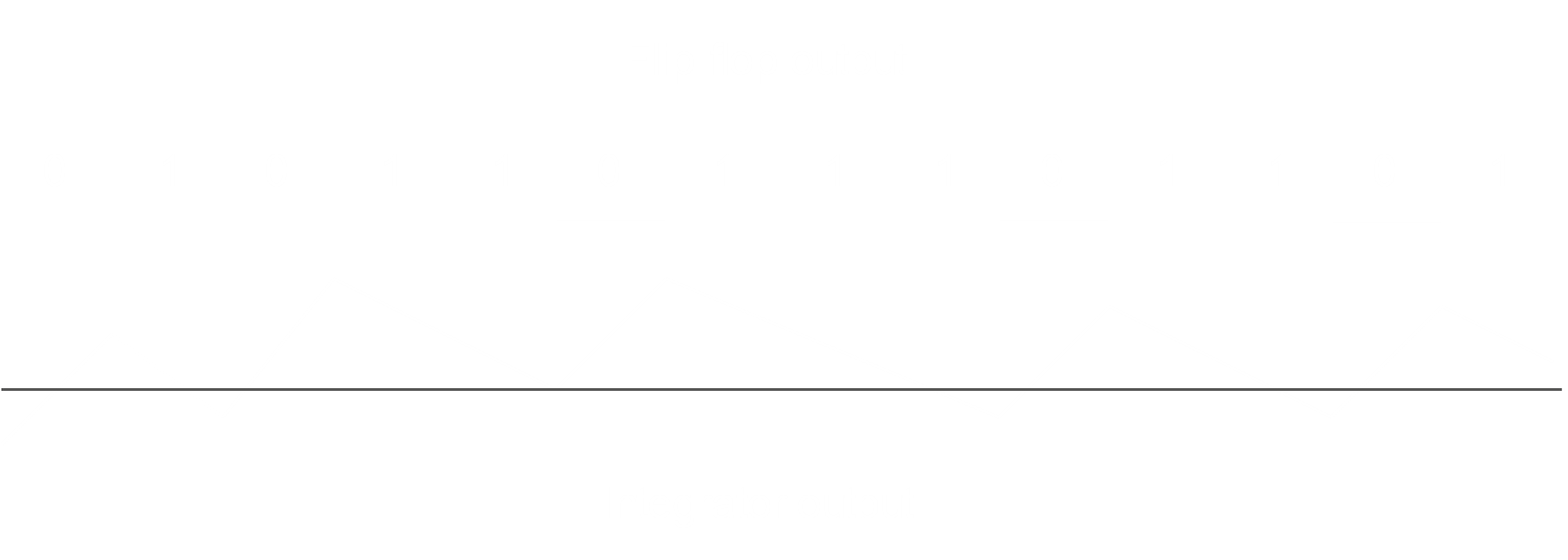


The **Delta-Sigma ADC** uses the **combinator circuit** shown above. The circuit works **bit-wise**, so there is no need to change any components if we want a higher resolution. This is an advantage over the Flash ADC.

Essentially, we **oversample** the input signal at **twice its rate** and assign a 0 or a 1 depending on whether the value is increasing or decreasing.

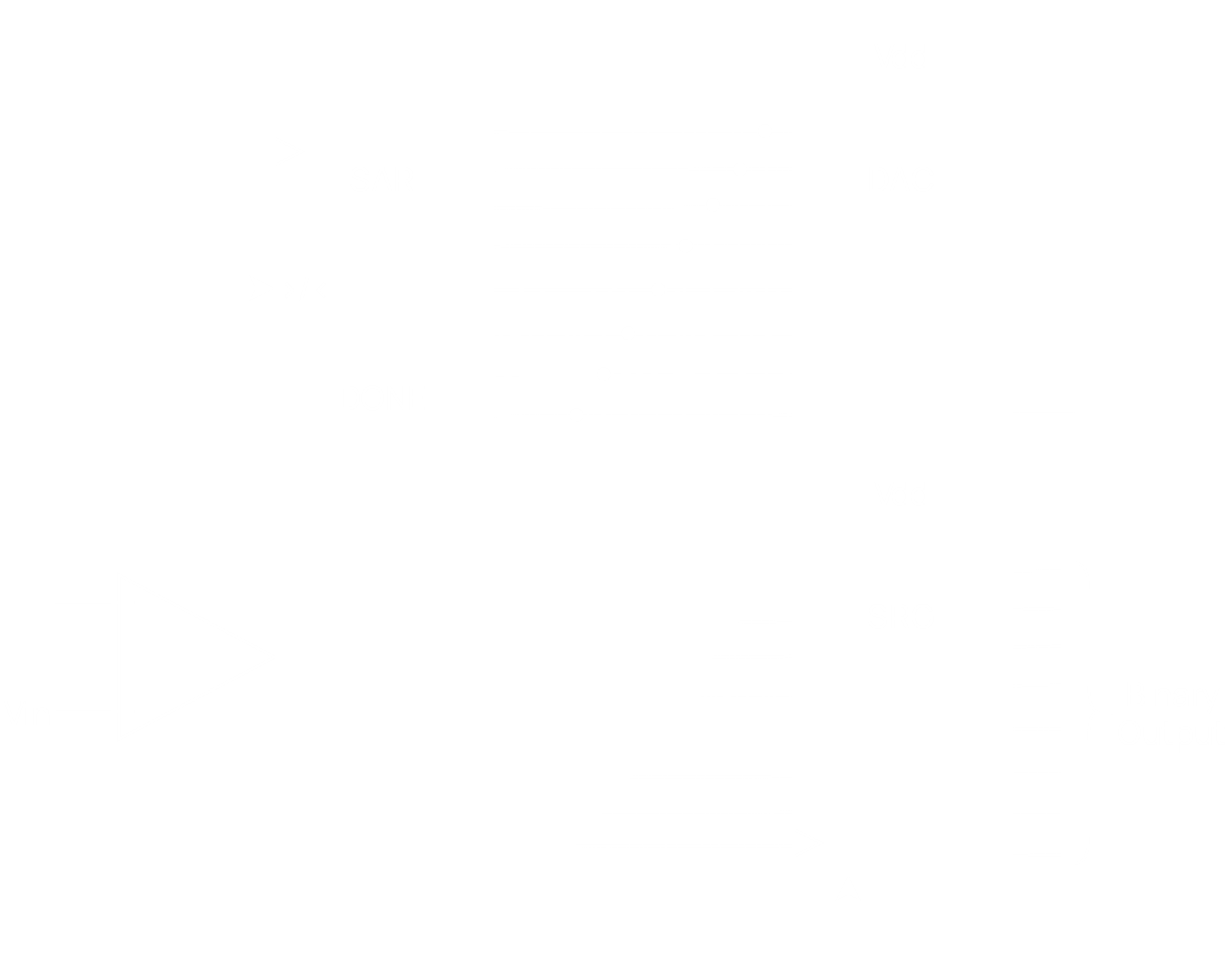






* High resolution
* No precision external components needed
* Accuracy is high
* Slow due to oversampling

## Successive Approximation ADC



A **Success Approximation ADC**, shown above, is used in modern computers. It is faster than the Delta-Sigma ADC but slower than the Flash ADC, while being more accurate than the Flash ADC but less accurate than the Delta-Sigma ADC.

Again, we have a **fixed circuit** with a **single comparator**, just like the Delta-Sigma ADC, and again we determine the output bit based on the voltage of the input signal compared to the reference voltage. However, the trick is that the **reference voltage changes**.

Consider that the input voltage, , at one specific moment of time is and that the initial reference voltage, , has been set to . These are the steps that are followed:

1. , so the left-most bit is
2. , so second left-most bit is
3. , so the third left-most bit is
4. , so the fourth left-most bit is

Thus, at each step, we set the value of using a formula. Notice the pattern being followed in the formula. If , then a bit is generated. Otherwise, a bit is generated. The bits are being generated from left to right, so we can keep going for as many bits as we need.

If we want a -bit output for example, for , the output will be . We can verify that this is correct.

Thus, we recreated the input voltage. This is done for every sample that is taken.

* Capable of high speeds while being reliable
* Medium accuracy compared to other ADCs
* Good trade-off between speed and accuracy
* Capable of serial output (one bit at a time)
* High resolution Successive Approximation ADCS are slow
* Less accurate than Flash ADCs

In conclusion,

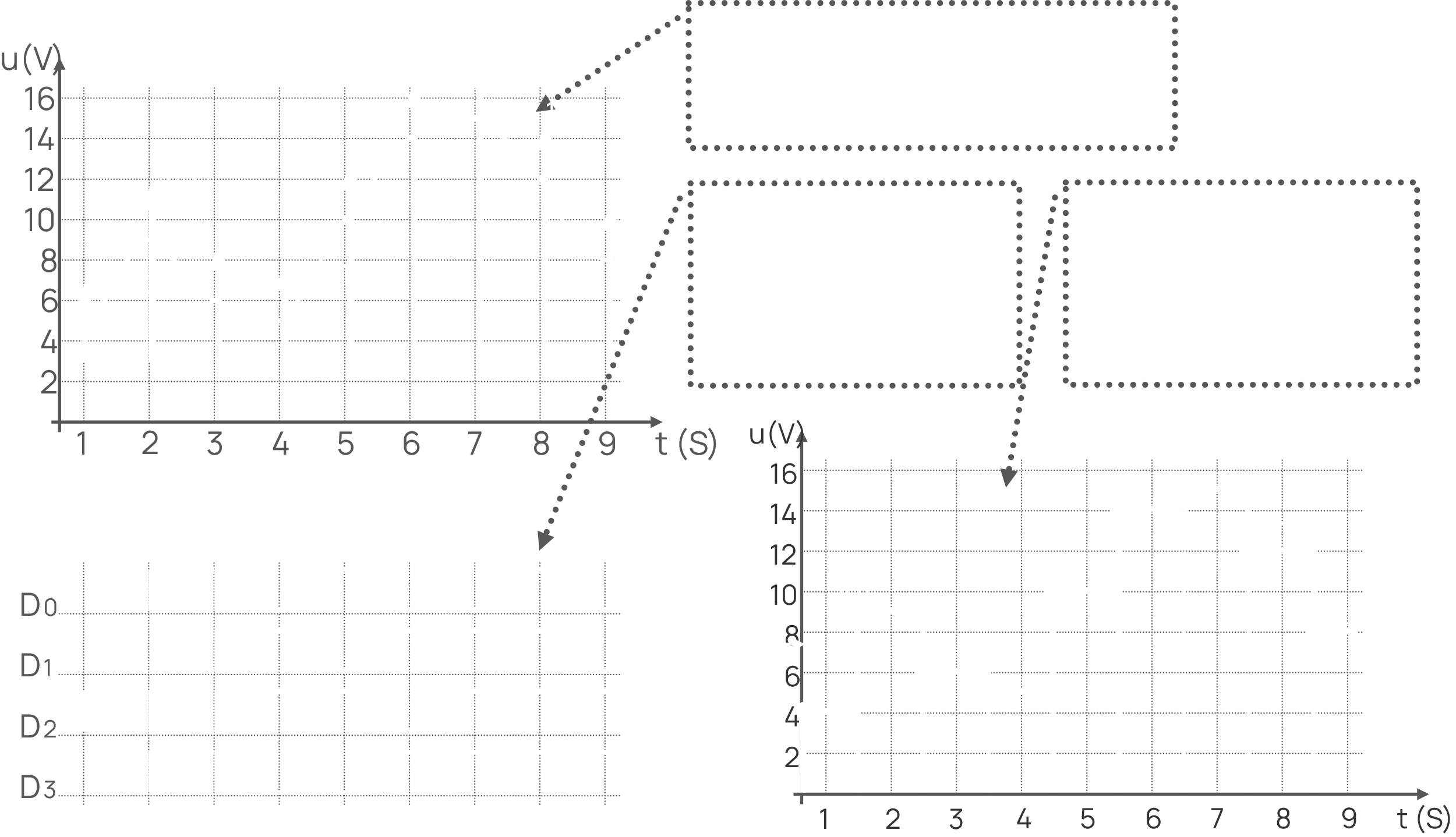
|  |  |  |
| --- | --- | --- |
|  | Speed (Relative) | Cost (Relative) |
| Flash ADC | Very Fast | High |
| Delta-Sigma ADC | Slow | Low |
| Successive-Approximation ADC | Medium/Fast | Low |

## Digital-to-Analogue Converters

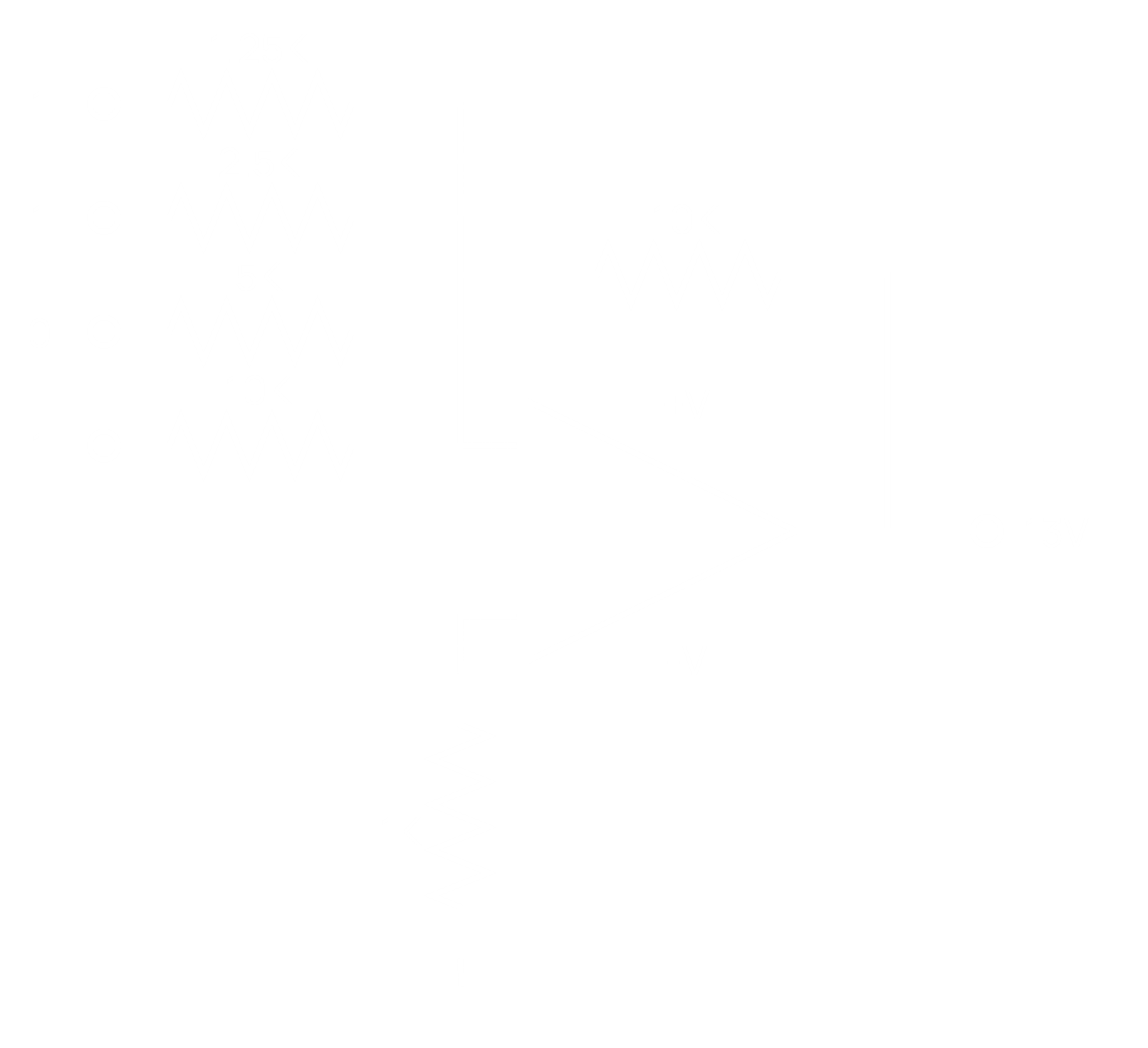
**Digital-to-Analogue Converters** simply take the digital value and output a specific voltage based on it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D3 | D2 | D1 | D0 | Vout (mV) |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 5 |
| 0 | 0 | 1 | 0 | 10 |
| 0 | 0 | 1 | 1 | 15 |
| 0 | 1 | 0 | 0 | 20 |
| 0 | 1 | 0 | 1 | 25 |
| 0 | 1 | 1 | 0 | 30 |
| 0 | 1 | 1 | 1 | 35 |
| 1 | 0 | 0 | 0 | 40 |
| 1 | 0 | 0 | 1 | 45 |
| 1 | 0 | 1 | 0 | 50 |
| 1 | 0 | 1 | 1 | 55 |
| 1 | 1 | 0 | 0 | 60 |
| 1 | 1 | 0 | 1 | 65 |
| 1 | 1 | 1 | 0 | 70 |
| 1 | 1 | 1 | 1 | 75 |

For an bit digital input, the output will be one of values. Because of this, the output signal will not be as accurate as the original analogue input, since the analogue input had an infinite number of values. Additionally, the values of the analogue signal changed constantly while those of the output from the DAC will only change at specific intervals.



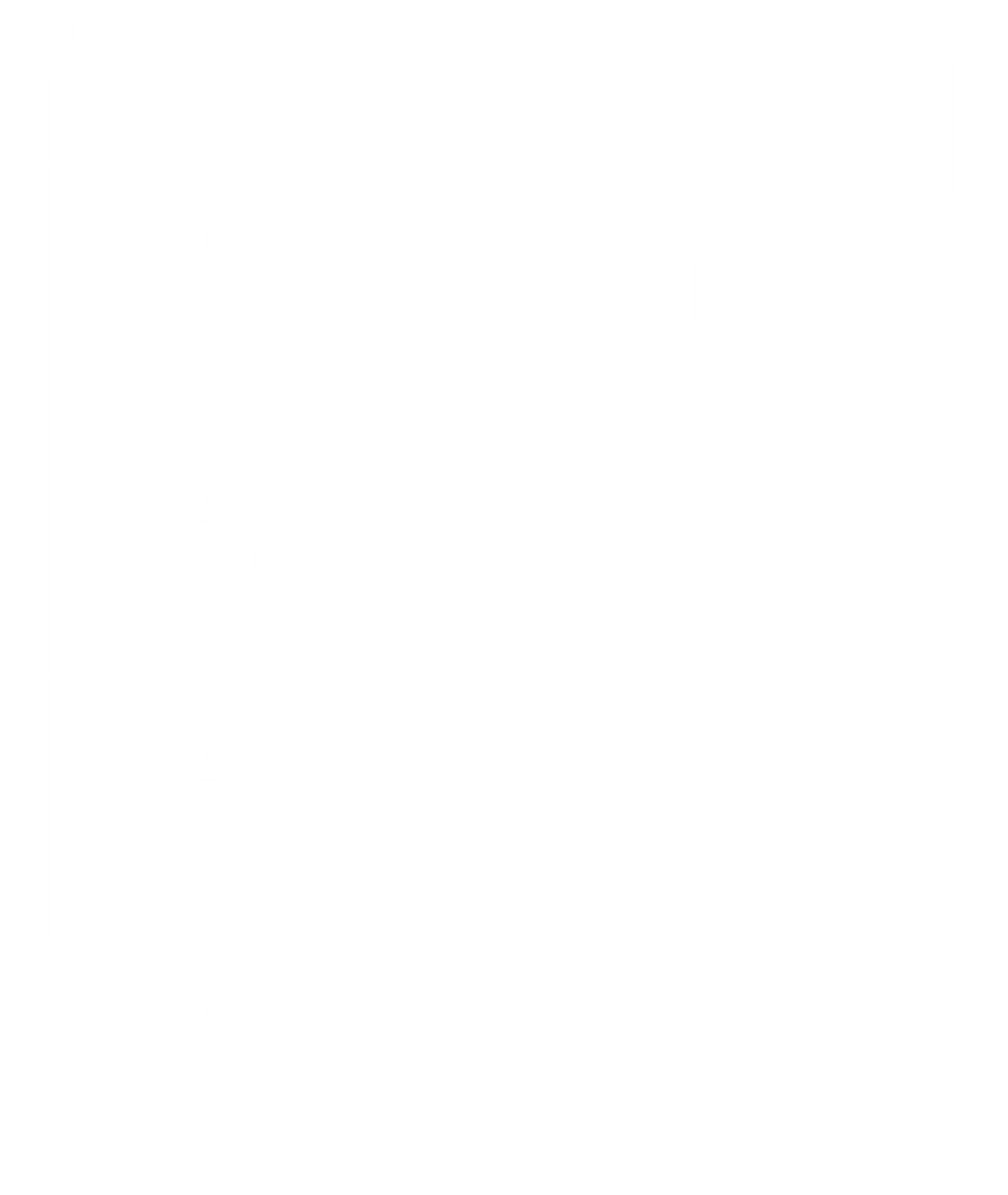
## Weighted Summing Amplifier DAC



As can be seen, in the **Weighted Summing Amplifier DAC**, we have a **separate register** connected to each bit of the digital input. The resistor values double for every bit from **left to right**. Whether the output value will be positive or negative depends on the circuit.

This approach is not all that good because for a large number of bits, the precision of the resistors will need to be very accurate.

## R-2R Ladder DAC



The **R-2R Ladder DAC** gets rid of the precision requirements of the resistors by having a simple setup. We only have two resistors, one with a value of and another with a value of for every bit. If we want to increase the number of bits, we simply add a module. The last group is different in that both resistors have a value of .

The final output voltage is found using this formula: