8-bit Digital to Analog converter (DAC)

Posted on February 28, 2008, by Ibrahim KAMAL, in General electronics, tagged

This article aims to introduce to beginners and intermediate readers a simple solution to build a digital to analog converter, based on the famous r/2r resistors network. This article also discuss a problem encountered by many beginners while trying to build their own DAC, and proposes some very simple solutions to that problem.

Learn how to build an Analog to digital converter using the same simple technique explained in this page.

Through this article, I am going to explain how to build an 8-bit digital to analog converter with parallel input. If you don't know what this means, well its simply a circuit that will take as input a digital 8-bit number from 0 (00000000) to 255 (11111111), and output the relative value on a scale from 0 to 5v.

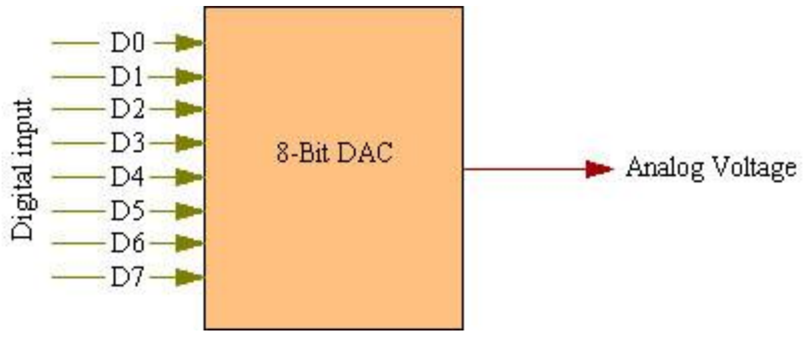
The maths that describe this process is very simple, an 8 bit converter will divide the 5 volts into 255 steps, each step having a value of:

5/255 = 0.019 V

Then the output voltage for the converter should be equal to the binary input multiplied by the step value, e.g. for an input of 129 (1000 0001 in binary) the output voltage should be:

129 X 0.019 = 2.451V

Here is a simplified functional diagram of an 8-bit DAC.



Some vocabulary:

DAC:Digital to Analog converter.

D0, D1, D..: Data lines.

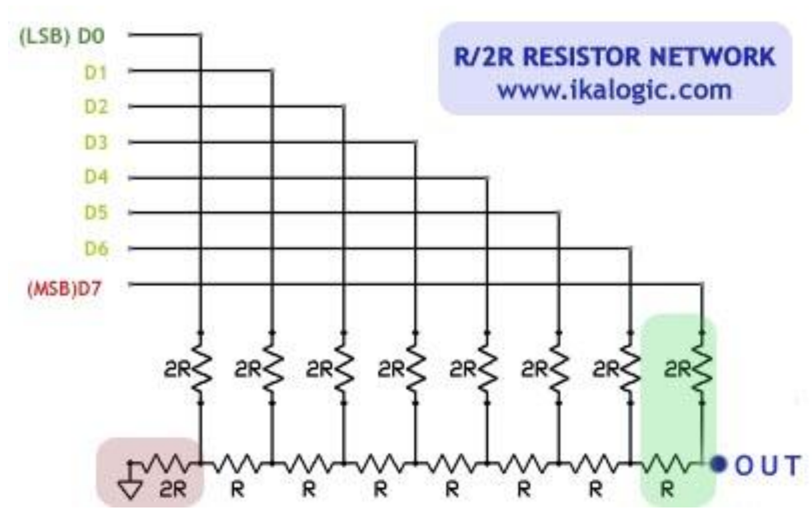
Analog: Continuous electrical signals.

Digital: Method of representing information using "1" and "0" (usually 5v and 0V).

LSB: Less significant bit.

MSB: Most significant bit.

The R/2R resistor network.



Follow the colors on the schematic and on the description text respectively, it can help!

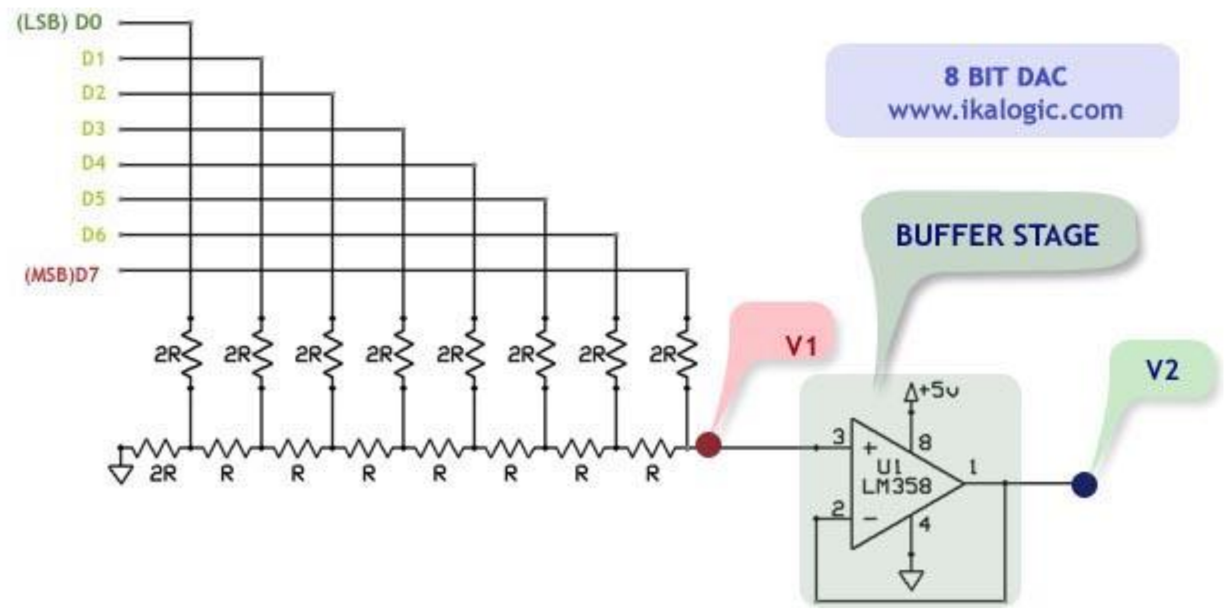
The digital data entering thought the 8 lines (**D_0 to D_7**) are going to be converted to an equivalent analog voltage (**V_{out}**) by the mean of the R/2R resistor network. Actually a lot of commercial Digital to Analog converter ICs are based on this same principle. The R/2R network is build by a set of resistors of two values, with one of them double the other (example 10K and 20K), in on of my circuits I used **1M ohm** and **470K ohm** resistors, which is quite near to the R/2R ratio, and this small difference didn't cause any detectable errors in most applications. However, if you want to build a very precise DAC, be precise when choosing the values of the resistors that will exactly match the **R/2R** ratio.

Note that you can build a DAC with any number of bits you want, simply by enlarging the resistor network, by adding more **R/2R** branches (like the one shaded in green), BUT you must keep the **2R** resistance connected to ground (shaded in light red)

Going through the mathematical proof for the operation of this converter can be a pain for some of us, and I am only intending to keep things simple.

Now, in order to use this Resistor Network (*also called R/2R Ladder*) for real applications, you will have to build a very simple voltage buffer circuit, which will be explained in the next section.

The applied circuit



Follow the colors on the schematic and on the description text respectively, it can help!
All the components are labeled on the circuit, so i'll start directly to explain how it works. To simplify this task, i'll split the circuit into two main stages: **the Digital to analog converter** and **the Voltage buffer stage**.

Stage 1: the Digital to analog converter (The R/2R network)

This part have been explained in detail in the previous section, its purpose is to create the voltage **V_1** which is equivalent to the weight of the binary number on the lines (**D_0 to D_7**). Now that this is a resistor network, if we apply any load on the output of the **first stage**, this load will be considered as an additional resistor in the network, and thus will disturb the network which will no longer provide the correct & desired output voltage. Therefore, to overcome this problem, we need a voltage buffer, here is where the next stage comes…

Stage 2: the voltage buffer

This stage will isolate the point **V_1** from the final output **V_2**, while always keeping the voltage **V_2** at the exact same value of **V_1**. This is what we call a voltage buffer. for the voltage buffer we use an opamp with the output connected to the inverting input (*this special configuration of the Op Amp is also called **Voltage Follower***). The most important things to note are:

1. No current (almost 0A) will flow from the point **V_1** into the OpAmp, so we wont be disturbing the resistor network configuration.
2. **V_2** will always equal **V_1** (theoretically, see the rest of this document).
3. The current going out from the point **V_2** to any other stage is sourced from the power supply of the OpAmp.

The most encountered problem & some solutions

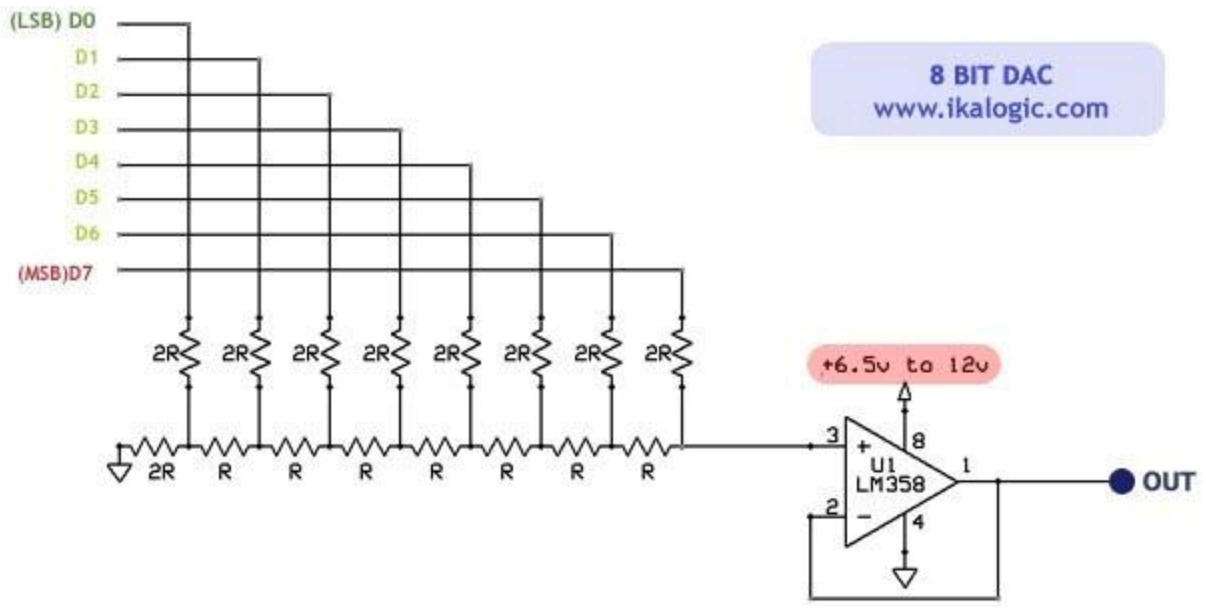
A quick look on those two graphs can be sufficient to understand the problem: the output of the op-amp is not linear on the full 0-to-Vcc scale. actually an OpAmp, depending on its type, will deliver a maximum voltage of (Vcc – 0.5V), where Vcc is the supply voltage of the OpAmp. So, in our application, the OpAmp will only deliver 4.5V even if theoretically it should deliver 5V.

You may think this caused by the resistor network, but it's not! this is a limitation in the op-amp itself.

Lets get a little deeper into the problem, the actual output curve in **red** should be linear, but actually it begins loosing its linearity beginning from 3.9 volt. (*Again this depends on the type of OpAmp, those results a based on my own tests on a LM350 OpAmp*) The red 'Error zone' is where the output of the DAC no longer math the relative binary input.

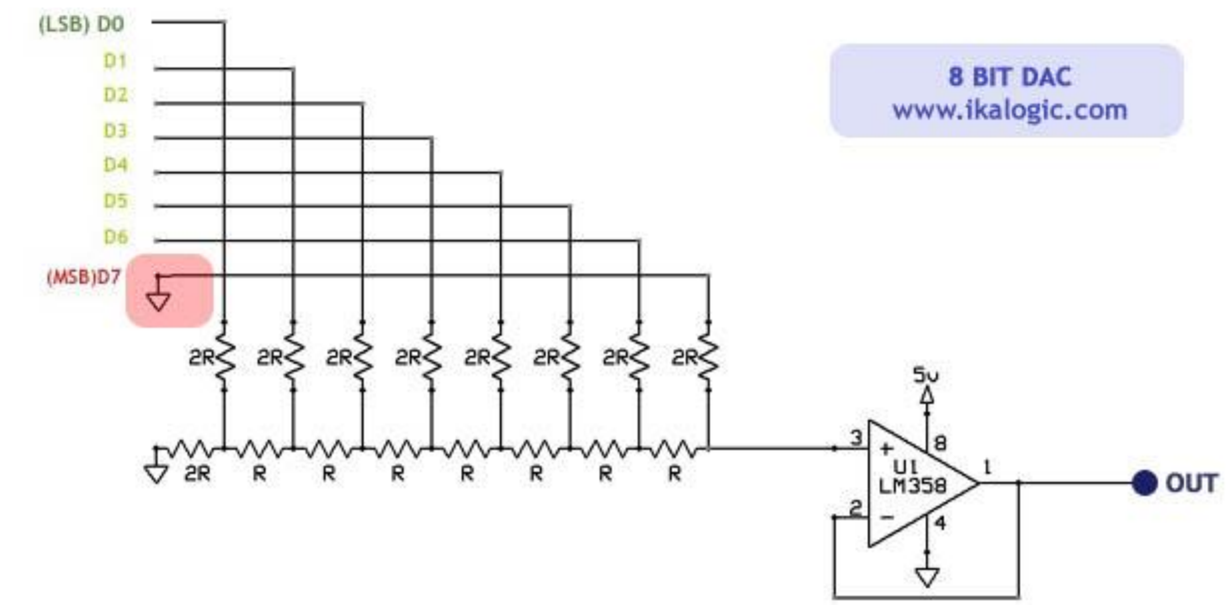
This is the error we will be trying to overcome in the next part, through two very simple solutions.

Solution 1 :



The first solutions – shown in the red shading – is to increase the supply voltage of the Op-Amp, as shown in the schematic. This will totally solve the problem, and, whether you are supplying 6.5 volts or more, you will get neat linear output from 0V to 5V.

Solution 2 :



The second solutions – shown in the red shading – is to reduce the range of the input to [0 to 127] from the original range of [0 to 255]. This will result on a voltage swing of 0 to 2.5 volt at the output, which will be in the linear operating area of the Op-Amp (*this done by attaching the MSB line to ground, this way you only control the 7 other lines, and a 7 bit value can swing from 0 to 127*).

I hope this introduction was interesting, and that it will help you to build simple but reliable DACs to suit your application.

Source: http://www.ikalogic.com/8-bit-digital-to-analog-converter-dac/