When we discussed digital to analog conversion (D to A), we mentioned that D to A was the simpler process, and that one of the better known A to D conversion processes actually relies upon the DAC as one of its components. Now that we've been introduced to the comparator, a device that compares two voltages to indicate which is more positive, we have the tools necessary to tackle A to D conversion.

Approaches to A to D Conversion

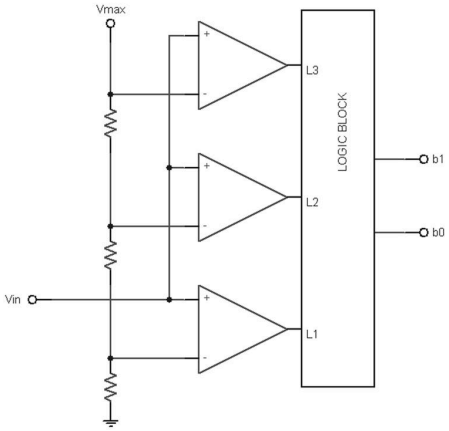
There are a number of different ways in which analog characteristics from the real world are digitized. The three main ways are:

1. Direct Conversion (also called Flash Conversion)
2. Fast Ramp Conversion
3. Successive Approximation Conversion

There are variations of the second one (Fast Ramp Conversion) which attempt to speed it up (contrary to its promising name, it is definitely the slowest of the approaches!) These are called things like Delta Conversion, Sigma Conversion, Delta-Sigma Conversion, etc. For this course, however, we will stick to the three main techniques.

Direct Conversion

Direct Conversion produces a binary result as a single event, or, in digital logic terms, in one clock cycle. As a result, this is a very fast technique, but the down-side to it is that it is very hardware intensive. Here's a simple 2-bit direct converter. (This is a pretty lousy A to D converter, as it only has four levels and three steps -- you're definitely not going to be listening to Tchaikovsky with this one!)



You'll get to build a 3-bit version of this circuit in the Lab, so instead of building and testing this circuit, we'll simply analyze it.

If V_{max} is 5.000 V and all three resistors are the same, we can analyze the thresholds for the comparators using a 3-resistor voltage divider as follows.

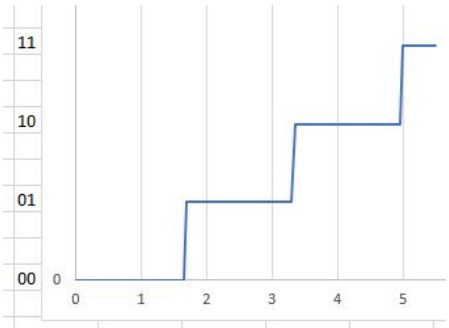
The current through the voltage divider would be

$I = \frac{V_{\max}}{3\ R}$, since the total resistance is the series combination of the three resistors.

The voltage for "L1" would then be

$$V_{L1} = R\left(\frac{V_{\max}}{3\ R}\right) \ = \ \frac{5}{3} = 1.667\ V$$

Following the same logic, the threshold for "L2" would be 3.333 V, and the threshold for "L3" would be 5.000 V. Here, then, is the relationship between the input voltage and the output binary code, given that we simply need to count the steps in binary:



Any voltage below 1.667 V is considered as 00. Any voltage between 1.667 V and 3.333 V is considered as 01. Any voltage between 3.333 V and 5.000 is considered as 10. Any voltage above 5.000 V is considered 11. This example really demonstrates that digitizing an analog signal can result in the loss of a lot of the original data! That's what we called "Quantization Error".

The "Logic Block" for this circuit needs to take the three comparator outputs and convert them into binary. Typically, this would be a Priority Encoder -- a circuit that generates a binary combination that represents the highest active input. By the way, all the lower inputs will also be active, because the input voltage is high enough to trigger them all.

This example is deceptively simple. Increasing the number of bits in the output increases the resolution of the circuit, but with an exponential increase in the hardware complexity. Notice that two bits were generated by four different voltage levels, three of which were supplied by comparators and the fourth was zero.

Let's consider what would be needed for more complex Direct Converters.

If we wanted three bits, we would need:

8 levels

7 comparators (one for each step), and a

7 input 3 output priority encoder.

Now, repeat for a sixteen bit Direct Converter.

65536 levels

65535 comparators and a logic circuit with

65535 inputs and 16 outputs. I'd like to see that circuit on a breadboard (or maybe a thousand breadboards!)

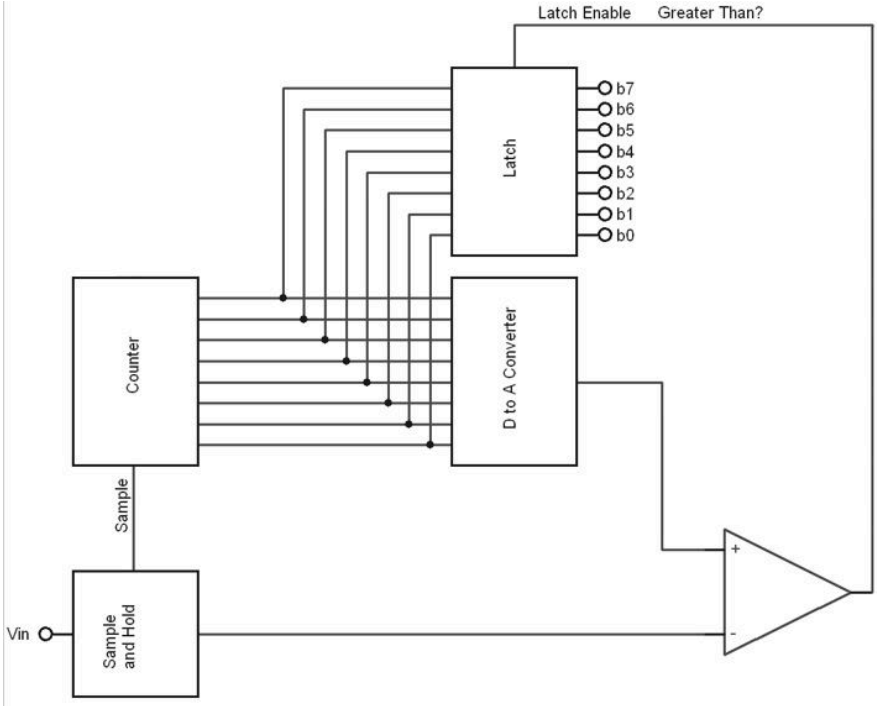
So, clearly, Direct Conversion is fast -- just one clock cycle -- but incredibly hardware intensive; and we haven't even talked about the digital logic involved in creating a Priority Encoder!

Fast Ramp Conversion

Clearly, there must be less hardware-intensive ways of digitizing characteristics from the real world. The Fast Ramp Converter is actually really slow. The name means you have to have a really fast binary ramp counter to make this even feasible.

With the Fast Ramp Converter, a DAC driven by a binary counter generates a constanty ramping output voltage until it reaches the top, then starts over again. The analog input signal is presented to one input of a comparatar, and the ramp is presented to the other input. When the ramp passes the analog signal, the value in the binary counter is stored as the binary representation of the analog input.

This introduces the need for a circuit called "Sample and Hold". At the beginning of each new ramp, a circuit at the input of the A to D converter grabs the present value of the input and keeps that value until the end of the ramp -- essentially an analog memory device. This is usually a capacitor, and the "switch" that temporarily connects the analog signal to it is usually a FET. The signal to get a new sample could be generated using the "Carry Out" of the counter. Here's the resulting block diagram of a Fast Ramp Converter.



In order for the sample rate to be constant (a definite requirement for proper digitization and playback), the Fast Ramp Converter needs to go through the entire counter ramp for each sample.

Just to see how slow this system is, let's compare it to a Direct Converter. Let's say that the clock running each of these devices issues a pulse every nanosecond.

How many samples per second would an 16-bit Direct Converter be able to make, in megasamples per second? 1000 MSa/s

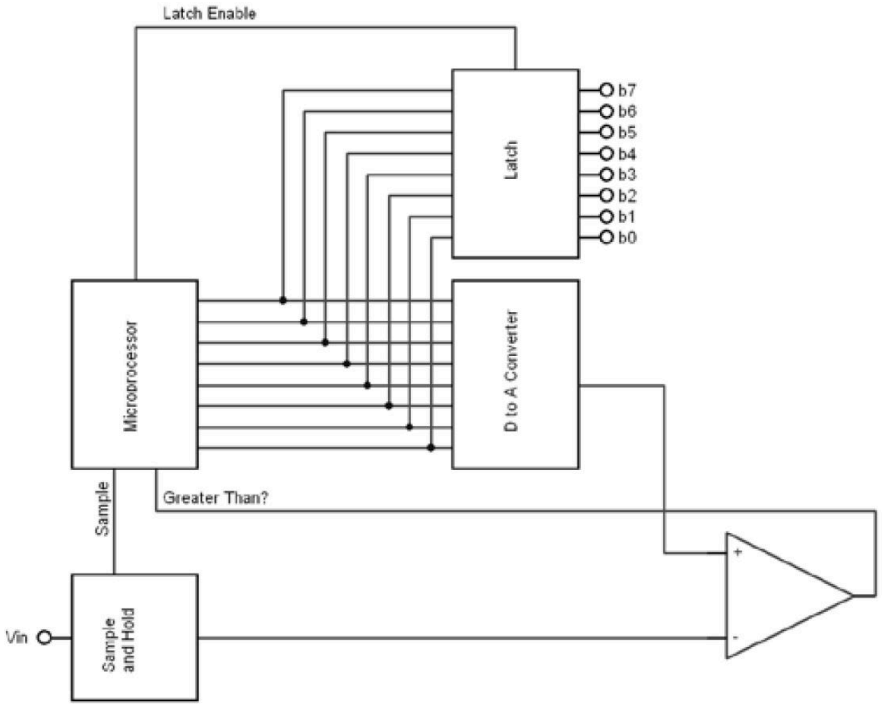
How many samples per second would an 16-bit Fast Ramp Converter be able to make, again in megasamples per second? (Start by determining how many different voltage levels there are in a 16-bit ramp.) 0.0152 MSa/s

That's significantly slower.

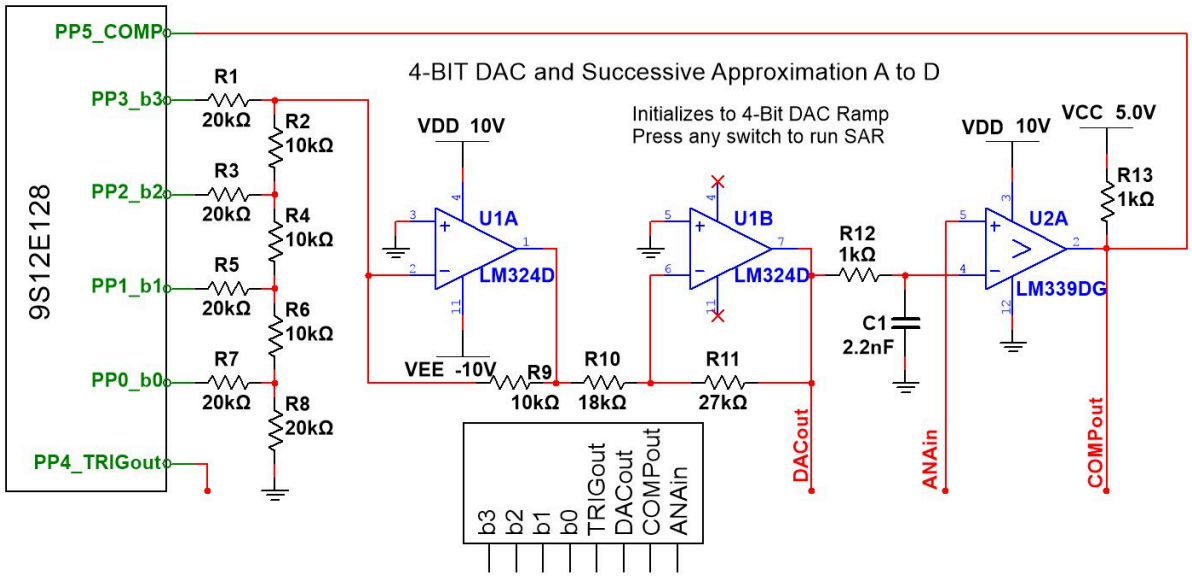
Successive Approximation

The happy medium between the two is a process called Successive Approximation, which is a smart guessing game. You could say that the Fast Ramp approach is a highly untelligent guessing game: Is it 0? No? Is it 1? No? Is it 2?, No? is it 3? Yes? Great; is it 4? No? is it 5? No? ....

The smarter way to go would be to split the field in half and pick the one with the answer in it, then split that half into quarters, etc. That's how a Successive Approximation A to D Converter works. The hardware looks pretty similar to the Fast Ramp, except for the addition of some "brains". By the way, a Successive Approximation A to D Converter is often referred to as an SAR, short for Successive Approximation Register, since the results are presented to the data bus for storage using a latching register.



Here's a 4-bit Successive Approximation DAC built up on one of the microcontroller boards from a few years ago.



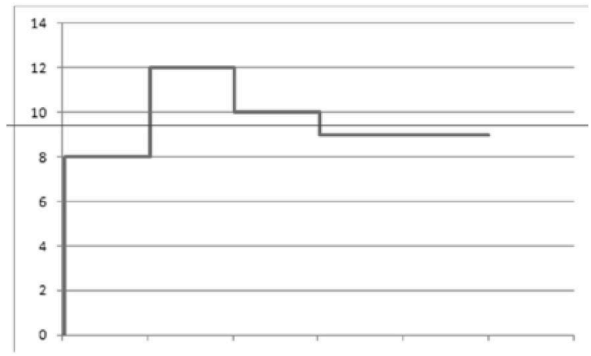
For this simple application, the Sample and Hold Circuit was left out, an omission which introduces significant error at higher frequencies.

The microprocessor on the left can generate any of the binary values from 0000 to 1111, which it sends into the R-2R ladder at the front end of the DAC. The R-2R ladder feeds into an inverting summing amplifier followed by a second inverting amplifier to generate a positive output. Next is a single-pole Low Pass Filter to reduce switching noise. After that comes the comparator. One of its inputs is the Analog signal, the other input is the DAC output. The comparator sends its binary output back to the microprocessor so it can make decisions based on whether the analog signal is greater or less than the DAC output.

For the fun of it, and for review, let's determine the step size. If all the outputs except the MSB are zero, then 5.0 V is presented to the summing amplifier through a 20 kΩ resistor. Since the feedback resistor is 10 kΩ, this means that the output of U1A will be -2.5 V. Multiplying that by the gain of the second amplifier, we calculate an output voltage of 3.75 V. Since this represents binary 1000 (only the MSB set), we divide by eight to get a step size of 468.75 mV/step. Bizarre value, but we'll live with it.

Successive Approximation Process

To simplify the analysis, let's instead start with an A to D converter with a step size of 1.0 V/step. The following picture shows the process of digitizing an input of 9.437 V.



To begin with, the microcontroller sets just the MSB to produce the binary code 1000 and, given a step size of 1 V/step, an output voltage of 8.0 V. Since this output is lower than the analog signal, the microcontroller keeps the MSB.

Now the microcontroller sets the next bit to produce the binary code 1100 and an output voltage of 12.0 V. Since this output is higher than the analog signal, the microcontroller clears this next bit.

Now the microcontroller sets the second last bit to produce the binary code 1010 and an output voltage of 10.0 V. Again, this output is higher than the analog signal, so the microcontroller clears this bit.

Finally, the microcontroller sets the LSB to produce the binary code 1001 and and output voltage of 9.0 V. Since this output is lower than the analog signal, the microcontroller keeps this bit, and presents its final guess as binary 1001. When played back by a DAC, this value will produce an output of 9.0 V instead of the original 9.437 V, again losing accuracy due to Quantization Error.

Let's compare the speed of a 16-bit Successive Approximation A to D Converter to the two previous converters, again assuming a clock pulse of 1 ns duration.

The Direct Conversion A to D could produce 1 billion samples in a second, or 1000 MSa/s

The Fast Ramp A to D took 65,536 clock cycles per sample, resulting in a pathetic 0.015 MSa/s

The Successive Approximation A to D takes 16 clock cycles for 16 bits, resulting in a rate of 62.5 MSa/s.

It's not super-speedy like the Direct Converter, but also it doesn't need a circuit with over 60,000 comparators and over 60,000 logic inputs; and it's definitely much faster than the Fast Ramp,with hardly any more circuit complexity.

Here are a couple of questions to finish up.

Match each of the following descriptions to the most likely 8-bit A to D converter.

2 Digital output generated after eight conversion steps

1 Digital output generated after one conversion step

3 Digital output generated after 256 conversion steps

1. Direct Conversion
2. Successive Approximation
3. Fast Ramp Conversion

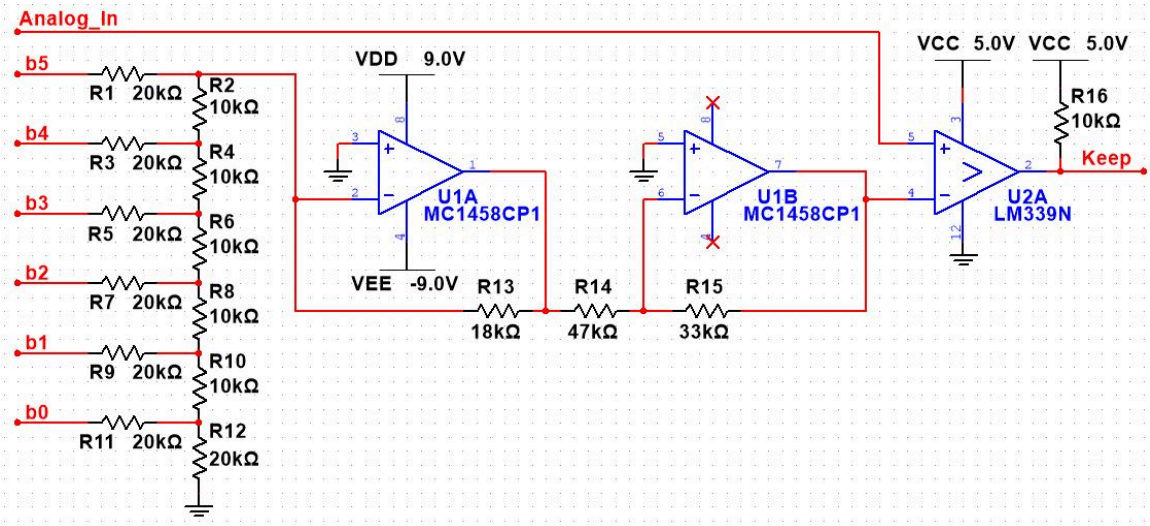
How many conversion steps would be required for a 12-bit A to D converter using each of the technologies listed?

Direct Conversion: 1

Fast Ramp Conversion: 4096

Successive Approximation: 12

Worked Example -- Successive Approximation A to D



We'll start with an overview of the circuit:

- There are six digital inputs, all of which are driven by 3.3 V LVTTL signals from a microcontroller (not shown)
- With six digital inputs, there are 2^6-1 steps, or 63 steps
- The inputs are handled by an R-2R ladder DAC, which should therefore be a linear binary-weighted DAC, with b5 as the MSB
- The two op amps in the DAC are both inverting, so the output voltage will always be zero or positive; hence, there is no need to clamp the input to the comparator
- When the analog signal is greater than the DAC output, the most recent "guess" should be kept; given that the analog input is connected to the non-inverting input, this means that a high output from the comparator indicates "keep"

To determine the step size for an R-2R ladder DAC, we set the MSB and clear all the rest, giving us a binary 100000, or decimal 32. In this state, none of the lower bits contribute any current to the circuit's input, so we can determine the expected output for 32 using the gains of the two amplifiers:

$$V_{32} = 3.3\left(-\frac{18\ k\Omega}{20\ k\Omega}\right)\left(-\frac{33\ k\Omega}{47\ k\Omega}\right) = 2.085\ V$$

The step size, therefore, is

$$Q = \frac{V_{32}}{32} = 65.2\ mV/step$$

The maximum output for 63 steps (binary 111111) would be 4.105 V

It's likely that the intent was for this DAC to have a range from 0 V to 4.095 V, and a better design would include a potentiometer in the op amp feedback to allow for some adjustment. For simplicity, let's work with the ideal step size:

Q=65 mV/step

From this, the voltage for binary 100000 would be 2.080 V and the output for binary 111111 would be 4.095 V.

Let's watch the process for the A to D converter digitizing the analog value 1.8702 V:

Binary (Decimal)	Voltage	Comparator
100000 (32)	2.080 (too high)	Don't Keep
010000 (16)	1.040 (too low)	Keep
011000 (24)	1.560 (too low)	Keep
011100 (28)	1.820 (too low)	Keep
011110 (30)	1.950 (too high)	Don't Keep
011101 (29)	1.885 (too high)	Don't Keep

Therefore, the final result is binary 011100, or 28 decimal, which produced the value 1.820 V -- a voltage that's less than one step below the analog input value of 1.8702 V.

In this example, it's fairly clear that 011101 or 29 produces 1.885 V which is actually closer to the analog input than the final result arrived at by this A to D converter. It's for this reason that a more sophisticated A to D converter would add half a step to the input voltage, bumping it up to 1.903 V, in which case the last bit would have resulted in "Keep", and the final answer would have been binary 011101, or decimal 29.