Analogue Logic

This section is written assuming you've either read the digital logic section already or have a good understanding of digital logic.

I'll be introducing some new components and concepts required to make the circuits that operate analogue logic, not previously discussed; once we've grasped diodes and OpAmps we'll start looking at concepts, circuits and some [mostly non-essential] math.

- As discussed previously, in digital and Boolean logic, the truth values of variables may only be the integer values 0 or 1.
- Analogue logic, [more commonly called fuzzy logic in the wider world], is a form of 'many-valued' logic in which the truth value of variables may be any real number between 0 and 1.

- since we are dealing with electronics this is further confused because '1' in these cases is the maximum voltage in the circuit, so rather than working between 0 and 1 we actually are working with 0 and, for example, +/-12V.
- Fortunately i'm going to keep the math simple by using the somewhat more vague value of 100% rather than a specific voltage like +/-12V.

- Analogue logic processing devices allow us to do some very interesting operations on signals and opens up a wide range of tuning and refinement that would be impossible, or at least prohibitively complex, with digital systems.
- Analogue processing is also where you will find some very subtle effects beyond the boring TRUE / FALSE of digital processing.

ANDs ORs XORs and NOTs

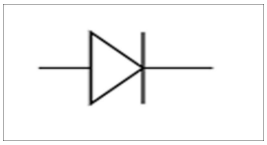
- These are our basic digital processing components, but they also have analogue processing abilities that are likely to be rather unexpected:

- The NOT simply passes the signal through, inverting it if required *(as discussed below, diode analogue logic has an inability to invert negatives)*
- The AND gate is effectively a MIN() function - it’s output is equal to the smallest value input.
- the OR gate is a MAX() function.
- The XOR is a strange case and seems to do something along the lines of a MAX() function on the analogue inputs, but then combines this with it’s binary output.

this means that to understand the complete behaviour you have to keep track of both analogue and digital inputs.

DIODES

- A **diode** is an electrical device allowing current to move through it **in one direction** with far **greater ease than** in **the other**.
- The most common kind of diode in modern circuit design is the semiconductor diode, although other diode technologies exist.
- Semiconductor diodes are symbolized in schematic diagrams as shown below.
- The term “diode” is customarily reserved for small signal devices, I ≤ 1A



Analogue Diode Logic

- Although these are simple as basic digital concepts, and true analogue versions of these concepts will be discussed, let's discuss briefly the slightly odd cases of some forms of NOT, AND and OR;
- When considering diode logic there are situations where it can be more than just digital 1 or 0, low or high but not a full true analogue:

- **A diode NOT simply passes the signal through, inverted**, but, since an inverted postitive will not pass through a reverse biased diode as far as logic is concerned, in diode logic **only negatives can be inverted by a NOT gate**, but those signals are full analogue.

- The **diode AND gate is a MIN() function** - it’s **output is equal to the smallest value input**, *not* multiplication as with true analogue.
- So, for example, for signals of 1.2V, 6.7V and 4.3V, the output would be 1.2V. this does however allow negative values through and also **is true analogue in a sense since it allows constant moving voltages of any value** up to the limits of the electronics.

- Likewise the **diode OR gate is a MAX() function** not an addition like true analogue, so the output value from the same three

signals would be 6.7V, again passing -ve values and continuous moving voltages in **true analogue** fashion is possible.

the above AND and OR for diode logic work as pure digital logic **in a true sense unlike the NOT**.

- Conversely, the **diode XOR gate** is even more odd and seems to do something along the lines of a MAX() function on the analogue inputs, but also sharing properties of a digital gate. That is, to understand the complete behaviour you have to keep track of both analogue and digital inputs, which means that **you can, in most situations use a diode XOR as a true analogue XOR** and **perform addition modulo2** like any other **ring mod or polarised VCA** etc.

- Most of the math in this document is based around diode logic circuitry so uses 100% of rail voltage as a reference maximum – e.g. in a eurorack modular system the math would be based on maximums of +/-12V.
- In fact I have generally ommitted the '%' symbol so when referring to 100 maximums you can read those calculations as +/-12V in place of the 100 in reality.

- Since a logical NOT gate mathematically can be said to perform:

1 – x

which is the same as

100% - x

for the most part, an inverted signal will be 100 – x, where x is the normal, uninverted value.

- This of course makes no sense with negative values of x, as the result would be greater than 100%.
- what actually happens with diode analogue logic is all analogue values only act as magnitude, before inversion, so:

```
      x   | inverted x
  ----------------------
      0   |    100
    100   |      0
     35   |     65
    -35   |     65
    -70   |     30
        etc.
```

[this will be explained in ore detail later on and specific cases where it doesn't hold true will mention this as and when necessary]

OPERATIONAL AMPLIFIERS

This section might seem a little heavier than most others as far as the science and math goes but it proves very useful in module design to understand these things well – it's not so essential as far as patching goes, even complex patch programming, so you could easily just read the summing up sections or maybe even skip it completely.

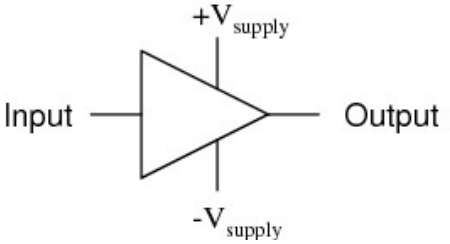
- The operational amplifier is arguably the most useful single device in analog electronic circuitry.
- With only a handful of external components, it can be made to perform a wide variety of analog signal processing tasks.
- It is also quite affordable, most general-purpose amplifiers selling for under a dollar apiece.
- Modern designs have been engineered with durability in mind as well: many "op-amps" are manufactured that can sustain direct short-circuits on their outputs without damage.

- One **key to the usefulness** of these little circuits is in the engineering principle of **feedback**, particularly ***negative*** **feedback**, which constitutes the foundation of almost all automatic control processes.

Single-ended and differential amplifiers

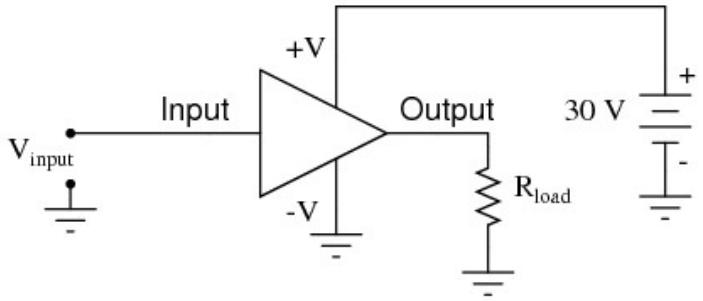
- For ease of drawing complex circuit diagrams, **electronic amplifiers** are often **symbolized by a simple triangle shape**, where the internal components are not individually represented.
- This symbology is very handy for cases where an amplifier's construction is irrelevant to the greater function of the overall circuit, and it is worthy of familiarization:

General amplifier circuit symbol



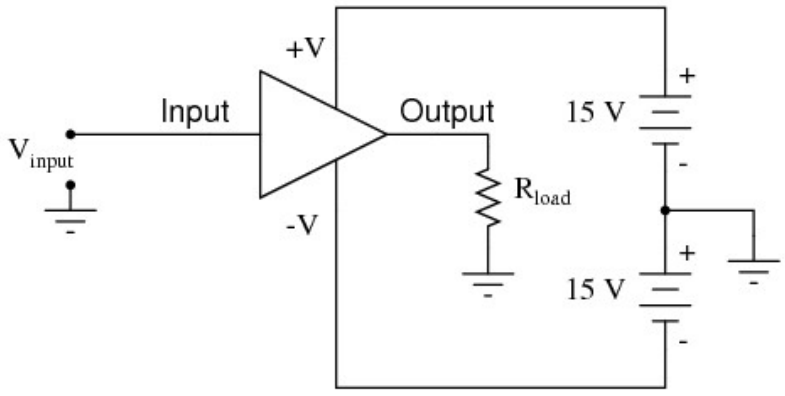
- The **+V** and **-V** connections denote the **positive** and **negative** sides of the **DC power supply**, respectively.

- The input and output voltage connections are shown as single conductors, because **it is assumed that all signal voltages are referenced to a common connection in the circuit called *ground***.
- Often (but not always!), one pole of the DC power supply, either positive or negative, is that ground reference point.
- A practical amplifier circuit (showing the input voltage source, load resistance, and power supply) might look like this:



- Without having to analyze the actual transistor design of the amplifier, you can readily discern the whole circuit's function: to **take an input signal** (V_{in}), **amplify it**, and **drive a load resistance** (R_{load}).
- To complete the above schematic, it would be good to specify the gains of that amplifier (A_V, A_I, A_P) and the Q (bias) point for any needed mathematical analysis.

- If it is necessary for an amplifier to be able to output true AC voltage (reversing polarity) to the load, a *split* DC power supply may be used, whereby the ground point is electrically "centered" between +V and -V.
- Sometimes the split power supply configuration is referred to as a *dual* power supply.

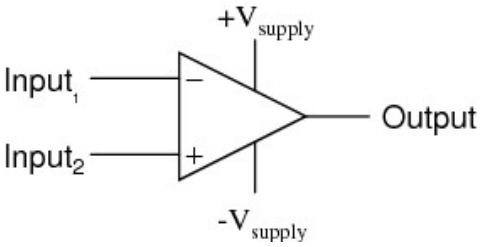


- The amplifier is still being supplied with 30 volts overall, but with the split voltage DC power supply, the output voltage across the load resistor can now swing from a theoretical maximum of +15 volts to -15 volts, instead of +30 volts to 0 volts.
- This is an easy way to get true alternating current (AC) output from an amplifier without resorting to capacitive or inductive (transformer) coupling on the output.

- The peak-to-peak amplitude of this amplifier's output between cutoff and saturation remains unchanged.

- By **signifying a transistor amplifier** within a larger circuit with a **triangle symbol**, we ease the task of studying and analyzing more complex amplifiers and circuits.
- One of these more complex amplifier types that we'll be studying is called the ***differential amplifier***.
- Unlike normal amplifiers, which amplify a single input signal (often called *single-ended* amplifiers), **differential amplifiers amplify** the **voltage difference between two input signals**.
- Using the simplified triangle amplifier symbol, a differential amplifier looks like this:

Differential amplifier



- The two input leads can be seen on the left-hand side of the triangular amplifier symbol, the output lead on the right-hand side, and the +V and -V power supply leads on top and bottom.
- As with the other example, **all voltages are referenced to the circuit's ground point**.
- Notice that one input lead is marked with a (-) and the other is marked with a (+).
- Because a differential amplifier amplifies the difference in voltage between the two inputs, each input influences the output voltage in opposite ways.
- Consider the following table of input/output voltages for a differential amplifier with a voltage gain of 4:

(-) $Input_1$	0	0	0	0	1	2.5	7	3	-3	-2
(+) $Input_2$	0	1	2.5	7	0	0	0	3	3	-7
Output	0	4	10	28	-4	-10	-28	0	24	-20

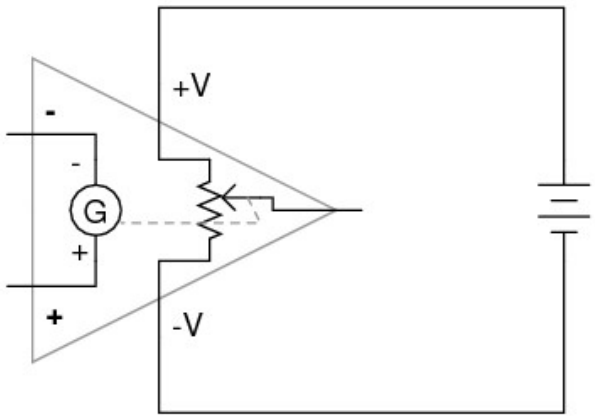
Voltage output equation: $V_{out} = A_V(Input_2 - Input_1)$

or

$V_{out} = A_V(Input_{(+)} - Input_{(-)})$

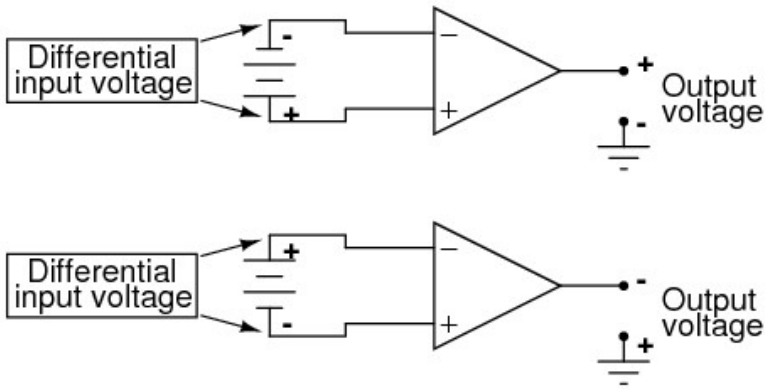
- An **increasingly positive voltage on the (+)** input **tends to drive the output voltage more positive**, and an **increasingly positive voltage on the (-)** input **tends to drive the output voltage more negative**.

- **Likewise**, an increasingly negative voltage on the (+) input tends to drive the output negative as well, and an increasingly negative voltage on the (-) input does **just the opposite**.
- Because of this relationship between inputs and polarities, the **(-) input** is commonly referred to as the ***inverting*** **input** and the **(+)** as the ***noninverting*** **input**.
- It may be helpful to think of a differential amplifier as a variable voltage source controlled by a sensitive voltmeter, as such:

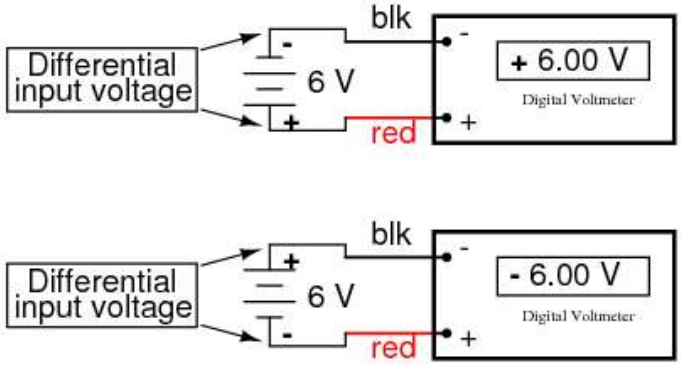


- Bear in mind that the above illustration is only a *model* to aid in understanding the behavior of a differential amplifier.
- It is not a realistic schematic of its actual design.
- The "G" symbol represents a galvanometer, a sensitive voltmeter movement.
- The potentiometer connected between +V and -V provides a variable voltage at the output pin (with reference to one side of the DC power supply), that variable voltage set by the reading of the galvanometer.
- It must be understood that **any load powered by the output of a differential amplifier gets its current from** the DC power source (battery), ***not*** **the input signal**.
- The input signal (to the galvanometer) merely *controls* the output.

- This concept may at first be confusing to students new to amplifiers.
- With all these polarities and polarity markings (- and +) around, its easy to get confused and not know what the output of a differential amplifier will be.
- To address this potential confusion, here's a simple rule to remember:



- **When the polarity of the *differential* voltage matches** the markings for inverting and noninverting inputs, the **output will be positive**.
- **When the polarity of the differential voltage clashes** with the input markings, the **output will be negative**.
- This bears some similarity to the mathematical sign displayed by digital voltmeters based on input voltage polarity.
- The red test lead of the voltmeter (often called the "positive" lead because of the color red's popular association with the positive side of a power supply in electronic wiring) is more positive than the black, the meter will display a positive voltage figure, and vice versa:



- Just as a voltmeter will only display the voltage *between* its two test leads, an **ideal differential amplifier only amplifies** the **potential difference between its two input connections**, not the voltage between any one of those connections and ground.
- The **output polarity** of a differential amplifier, just like the signed indication of a digital voltmeter, **depends on the relative polarities** of the differential voltage **between the two input connections**.

- If the input voltages to this amplifier represented mathematical quantities (as is the case within analog computer circuitry), or physical process measurements (as is the case within analog electronic instrumentation circuitry), you can see how a device such as a differential amplifier could be very useful.

- We could use it to compare two quantities to see which is greater (by the polarity of the output voltage), or perhaps we could compare the difference between two quantities (such as the level of liquid in two tanks) and flag an alarm (based on the absolute value of the amplifier output) if the difference became too great.
- In basic automatic control circuitry, the quantity being controlled (called the *process variable*) is compared with a target value (called the *setpoint*), and decisions are made as to how to act based on the discrepancy between these two values.
- The first step in electronically controlling such a scheme is to amplify the difference between the process variable and the setpoint with a differential amplifier.
- In simple controller designs, the output of this differential amplifier can be directly utilized to drive the final control element (such as a valve) and keep the process reasonably close to setpoint.

To Sum Up:
- A "shorthand" symbol for an electronic **amplifier** is a **triangle**, the wide end signifying the input side and the narrow end signifying the output. Power supply lines are often omitted in the drawing for simplicity.

- To facilitate true AC output from an amplifier, we can **use** what is called **a *split* or *dual* power supply**, with two DC voltage sources connected in series with the middle point grounded, giving a positive voltage to ground (+V) and a negative voltage to ground (-V). Split power supplies like this are frequently used in differential amplifier circuits.

- Most amplifiers have one input and one output. ***Differential amplifiers* have two inputs and one output**, the output signal being proportional to the difference in signals between the two inputs.

- The voltage output of a differential amplifier is determined by the following equation: $V_{out} = A_V(V_{noninv} - V_{inv})$

The "operational" amplifier

- Long before the advent of digital electronic technology, computers were built to electronically perform calculations by employing voltages and currents to represent numerical quantities.
- This was especially useful for the simulation of physical processes.
- A variable voltage, for instance, might represent velocity or force in a physical system.
- Through the use of resistive voltage dividers and voltage amplifiers, the mathematical operations of division and multiplication could be easily performed on these signals.

- The reactive properties of capacitors and inductors lend themselves well to the simulation of variables related by calculus functions.
- Remember how the current through a capacitor was a function of the voltage's rate of change, and how that rate of change was designated in calculus as the *derivative*? Well, if voltage across a capacitor were made to represent the velocity of an object, the current through the capacitor would represent the force required to accelerate or decelerate that object, the capacitor's capacitance representing the object's mass:

$$i_C = C \frac{dv}{dt}$$

Where,

i_C = Instantaneous current through capacitor

C = Capacitance in farads

$\frac{dv}{dt}$ = Rate of change of voltage over time

$$F = m \frac{dv}{dt}$$

Where,

F = Force applied to object

m = Mass of object

$\frac{dv}{dt}$ = Rate of change of velocity over time

- This analog electronic computation of the calculus derivative function is technically known as *differentiation*, and it is a natural function of a capacitor's current in relation to the voltage applied across it.
- Note that this circuit requires no "programming" to perform this relatively advanced mathematical function as a digital computer would.

aside: some might enjoy knowing you can build an opamp differentiator and here's the math for it;

The voltage output for the operational amplifier differentiator can be determined from the relationship below:

Vout =−R C dVin/dt
Where:
Vout = output voltage from op amp differentiator
Vin = input voltage
t = time in seconds
R = resistor value in the differentiator in Ω
C = capacitance of differentiator capacitor in Farads
dVin/dt = rate of change of voltage with time.

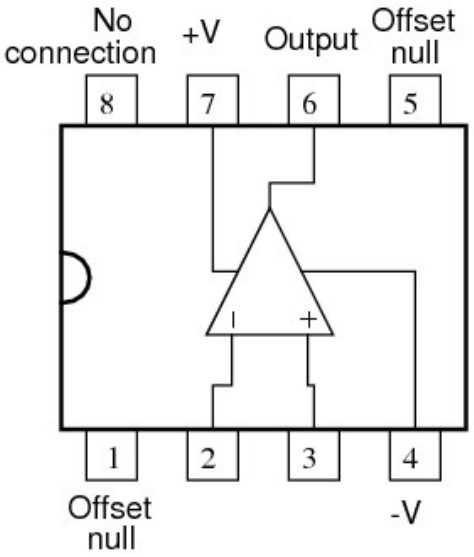
- Electronic circuits are very easy and inexpensive to create compared to complex physical systems, so this kind of analog electronic simulation was widely used in the research and development of mechanical systems.
- For realistic simulation, though, amplifier circuits of high accuracy and easy configurability were needed in these early computers.

- It was found in the course of analog computer design that differential amplifiers with extremely high voltage gains met these requirements of accuracy and configurability better than single-ended amplifiers with custom-designed gains.
- Using simple components connected to the inputs and output of the high-gain differential amplifier, virtually any gain and any function could be obtained from the circuit, overall, without adjusting or modifying the internal circuitry of the amplifier itself.
- These **high-gain differential amplifiers** came to be **known as** ***operational amplifiers***, or ***op-amps***, because of their application in analog computers' mathematical *operations*.

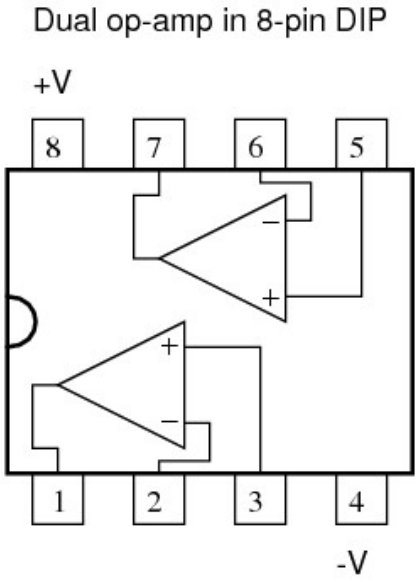
- Modern op-amps, like the popular model 741, are high-performance, inexpensive integrated circuits.
- Their **input impedances are quite high**, the inputs drawing currents in the range of half a microamp (maximum) for the 741, and far less for op-amps utilizing field-effect input transistors.
- **Output impedance is typically quite low**, about 75 Ω for the model 741, and **many models have built-in output short circuit protection**, meaning that their outputs can be directly shorted to ground without causing harm to the internal circuitry. With direct coupling between op-amps' internal transistor stages, they can amplify DC signals just as well as AC (up to certain maximum voltage-risetime limits).
- It would cost far more in money and time to design a comparable discrete-transistor amplifier circuit to match that kind of performance, unless high power capability was required. For these reasons, op-amps have all but obsoleted discrete-transistor signal amplifiers in many applications.

- The following diagram shows the pin connections for single op-amps (741 included) when housed in an 8-pin DIP (Dual Inline Package) integrated circuit:

Typical 8-pin "DIP" op-amp integrated circuit



- Some models of op-amp come two to a package, including the popular models TL082 and 1458.
- These are called "dual" units, and are typically housed in an 8-pin DIP package as well, with the following pin connections:

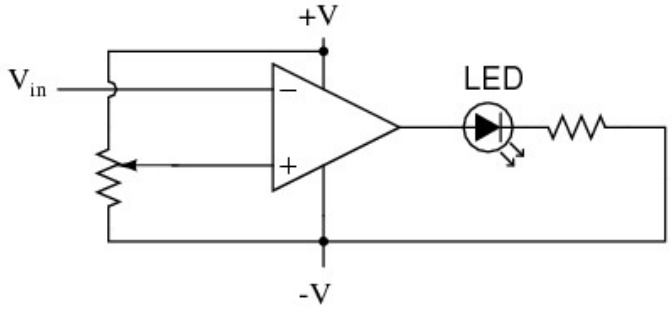


- Operational amplifiers are also available four to a package, usually in 14-pin DIP arrangements.
- Unfortunately, pin assignments aren't as standard for these "quad" op-amps as they are for the "dual" or single units.
- Consult the manufacturer datasheet(s) for details.

- Practical **operational amplifier voltage gains** are in the **range of 200,000** or more, which makes them **almost useless as an analog differential amplifier** by themselves.

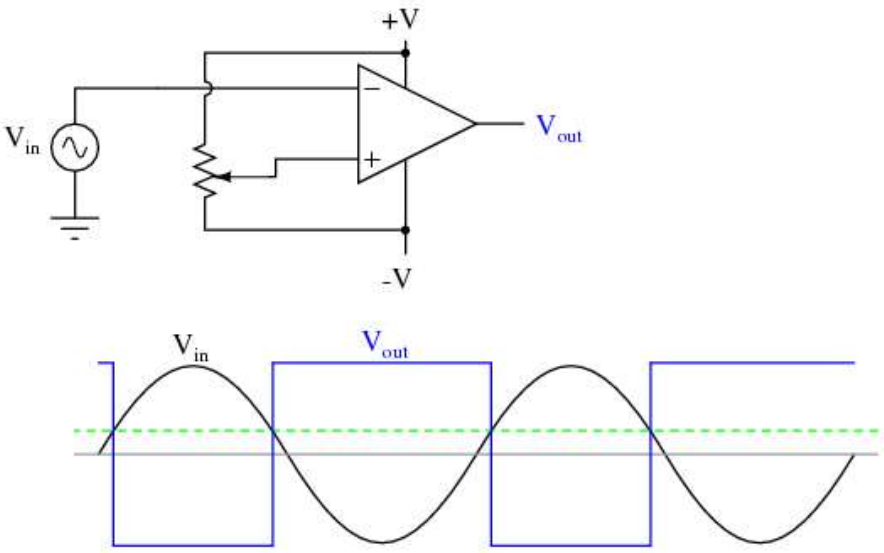
- For an op-amp with **a voltage gain (A_V) of 200,000** and a **maximum output voltage swing of +15V/-15V**, all it **would take** is a **differential input voltage of 75 µV** (microvolts) **to drive it to saturation or cutoff**!
- Before we take a look at how external components are used to bring the gain down to a reasonable level, let's investigate applications for the "bare" op-amp by itself.

- One application is called the ***comparator***.
- For all practical purposes, we can say that **the output of an op-amp will be saturated fully positive if the (+) input is more positive than the (-) input**, and saturated **fully negative if the (+) input is less positive than the (-) input**.
- In other words, an op-amp's extremely high voltage gain makes it useful as a device to compare two voltages and change output voltage states when one input exceeds the other in magnitude.

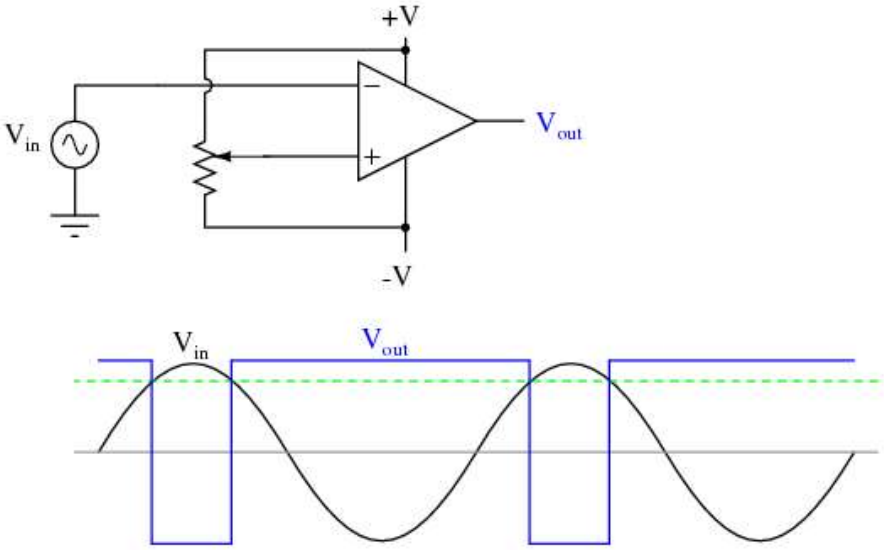


- In the above circuit, we have an op-amp connected as a comparator, comparing the input voltage with a reference voltage set by the potentiometer (R_1).
- If V_{in} drops below the voltage set by R_1, the op-amp's output will saturate to +V, thereby lighting up the LED. Otherwise, if V_{in} is above the reference voltage, the LED will remain off.
- If V_{in} is a voltage signal produced by a measuring instrument, this comparator circuit could function as a "low" alarm, with the trip-point set by R_1.
- Instead of an LED, the op-amp output could drive a relay, a transistor, an SCR, or any other device capable of switching power to a load such as a solenoid valve, to take action in the event of a low alarm.

- Another application for the comparator circuit shown is a square-wave converter.
- Suppose that the input voltage applied to the inverting (-) input was an AC sine wave rather than a stable DC voltage. In that case, the output voltage would transition between opposing states of saturation whenever the input voltage was equal to the reference voltage produced by the potentiometer.
- **The result would be a square wave:**

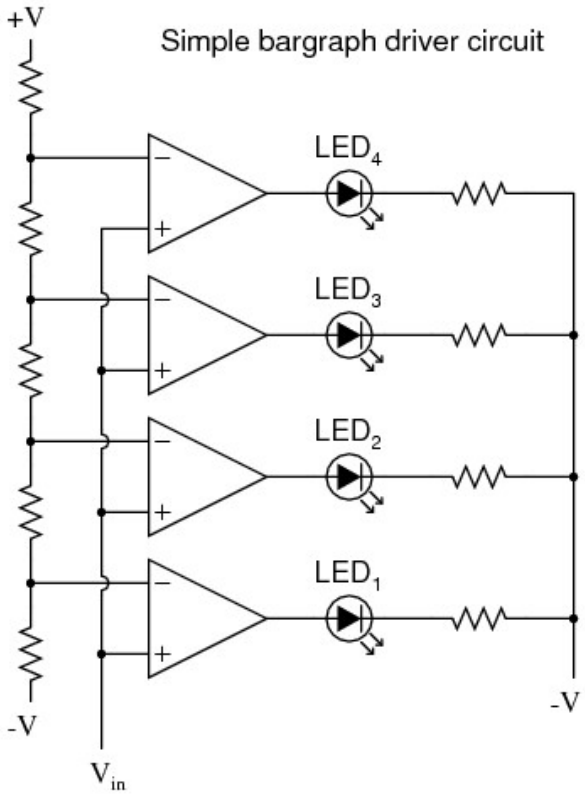


- Adjustments to the potentiometer setting would change the reference voltage applied to the noninverting (+) input, which would change the points at which the sine wave would cross, changing the on/off times, or *duty cycle* of the square wave:



- It should be evident that the AC input voltage would not have to be a sine wave in particular for this circuit to perform the same function.
- The input voltage could be a triangle wave, sawtooth wave, or any other sort of wave that ramped smoothly from positive to negative to positive again.
- This sort of comparator circuit is very useful for creating square waves of varying duty cycle.
- This technique is sometimes referred to as ***pulse-width modulation***, or **PWM** (varying, or *modulating* a waveform according to a controlling signal, in this case the signal produced by the potentiometer).
- Another comparator application is that of the bargraph driver.

- If we had several op-amps connected as comparators, each with its own reference voltage connected to the inverting input, but each one monitoring the same voltage signal on their noninverting inputs, we could build a bargraph-style meter such as what is commonly seen on the face of stereo tuners and graphic equalizers.
- As the signal voltage (representing radio signal strength or audio sound level) increased, each comparator would "turn on" in sequence and send power to its respective LED.
- With each comparator switching "on" at a different level of audio sound, the number of LED's illuminated would indicate how strong the signal was.



- In the circuit shown above, LED_1 would be the first to light up as the input voltage increased in a positive direction.
- As the input voltage continued to increase, the other LED's would illuminate in succession, until all were lit.
- This very same technology is **used in some analog-to-digital signal converters, namely the *flash converter***, to translate an analog signal quantity into a series of on/off voltages representing a digital number.

To Sum Up:
- A triangle shape is the generic symbol for an amplifier circuit, the wide end signifying the input and the narrow end signifying the output.

- Unless otherwise specified, ***all* voltages in amplifier circuits are referenced to a common *ground* point**, usually connected to one

terminal of the power supply. This way, we can speak of a certain amount of voltage being "on" a single wire, while realizing that voltage is *always* measured between two points.

- A ***differential amplifier*** is one **amplifying the voltage *difference* between two signal inputs**. In such a circuit, one input tends to drive the output voltage to the **same polarity of the input signal**, while the **other input does just the opposite**. Consequently, the **first input is called the *noninverting* (+)** input and the **second is called the *inverting* (-) input**.

- An *operational amplifier* (or ***op-amp*** for short) is a **differential amplifier with an extremely high voltage gain** (A_V = 200,000 or more). Its name hails from its original use in analog computer circuitry (performing mathematical *operations*).

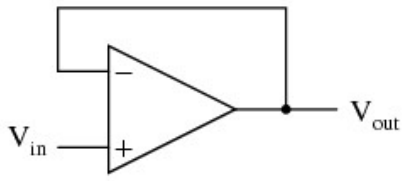
- Op-amps typically have very high input impedances and fairly low output impedances.

- Sometimes op-amps are used as signal ***comparators***, operating in full cutoff or saturation mode depending on which input (inverting or noninverting) has the greatest voltage. Comparators are useful in detecting "greater-than" signal conditions (comparing one to the other).

- One comparator application is called the ***pulse-width modulator***, and is made by comparing a sine-wave AC signal against a DC reference voltage. As the DC reference voltage is adjusted, the square-wave output of the comparator changes its duty cycle (positive versus negative times). Thus, the DC reference voltage controls, or *modulates* the pulse width of the output voltage.

Negative feedback

- If we **connect the output of an op-amp to its inverting input** and **apply a voltage signal to the noninverting input**, we find that the **output voltage** of the op-amp closely **follows that input voltage** (I've neglected to draw in the power supply, +V/-V wires, and ground symbol for simplicity):



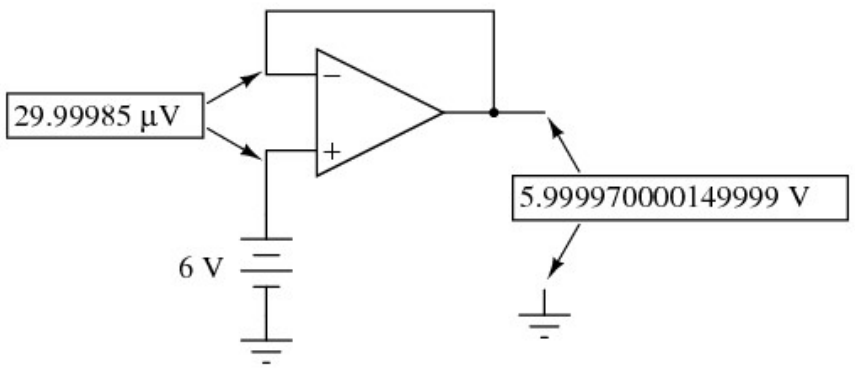
- As V_{in} increases, V_{out} will increase in accordance with the differential gain.

- However, **as V_{out} increases, that output voltage is fed back to the inverting input, thereby acting to decrease the voltage differential between inputs**, which acts to bring the output down. What will happen for any given voltage input is that the op-amp will output a voltage very nearly equal to V_{in}, but just low enough so that there's enough voltage difference left between V_{in} and the (-) input to be amplified to generate the output voltage.

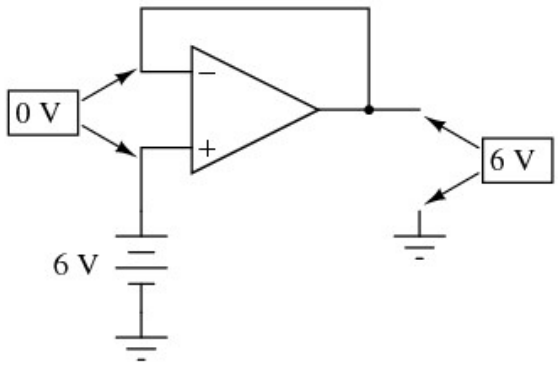
- **The circuit will quickly reach a point of stability** (known as *equilibrium* in physics), where the output voltage is just the right amount to maintain the right amount of differential, which in turn produces the right amount of output voltage.
- Taking the op-amp's output voltage and coupling it to the inverting input is a technique known as ***negative feedback***, and it is the key to having a self-stabilizing system (this is true not only of op-amps, but of any dynamic system in general).
- **This stability gives** the op-amp the capacity to work in its **linear (active) mode**, **as opposed to** merely **being saturated fully "on" or "off" as it was when used as a comparator**, with no feedback at all.

- Because the op-amp's gain is so high, the voltage on the inverting input can be maintained almost equal to V_{in}.
- Let's say that our op-amp has a differential voltage gain of 200,000.
- If V_{in} equals 6 volts, the output voltage will be 5.999970000149999 volts.
- This creates just enough differential voltage (6 volts - 5.999970000149999 volts = 29.99985 µV) to cause 5.999970000149999 volts to be manifested at the output terminal, and the system holds there in balance.
- As you can see, 29.99985 µV is not a lot of differential, so for practical calculations, we can assume that the differential voltage between the two input wires is held by negative feedback exactly at 0 volts.

The effects of negative feedback

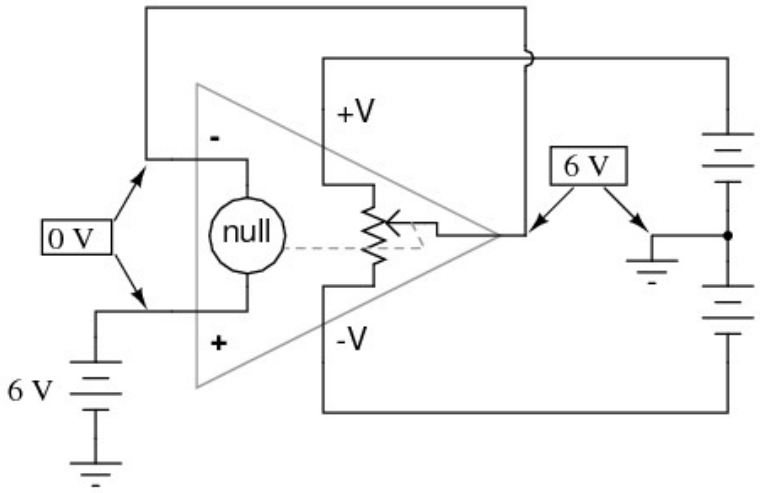


The effects of negative feedback
(rounded figures)



- One great advantage to **using an op-amp with negative feedback** is that the **actual voltage gain** of the op-amp **doesn't matter, so long as its very large**.
- If the op-amp's differential gain were 250,000 instead of 200,000, all it would mean is that the output voltage would hold just a little closer to V_{in} (less differential voltage needed between inputs to generate the required output).
- In the circuit just illustrated, the output voltage would still be (for all practical purposes) equal to the non-inverting input voltage.
- Op-amp gains, therefore, do not have to be precisely set by the factory in order for the circuit designer to build an amplifier circuit with precise gain.
- Negative feedback makes the system self-correcting.
- The above circuit as a whole will simply follow the input voltage with a stable gain of 1.

- Going back to our differential amplifier model, **we can think of the operational amplifier as being a variable voltage source controlled by an extremely sensitive *null detector***, the kind of meter movement or other sensitive measurement device used in bridge circuits to detect a condition of balance (zero volts).
- The "potentiometer" inside the op-amp creating the variable voltage will move to whatever position it must to "balance" the inverting and noninverting input voltages so that the "null detector" has zero voltage across it:



- As the "potentiometer" will move to provide an output voltage necessary to satisfy the "null detector" at an "indication" of zero volts, the output voltage becomes equal to the input voltage: in this case, 6 volts.
- If the input voltage changes at all, the "potentiometer" inside the op-amp will change position to hold the "null detector" in balance (indicating zero volts), resulting in an output voltage approximately equal to the input voltage at all times.

- This will hold true within the range of voltages that the op-amp can output.
- **With a power supply of +15V/-15V**, and **an ideal amplifier** that can swing its output voltage just as far, it **will faithfully "follow" the input voltage** between the limits of **+15 volts** and **-15 volts**.
- For this reason, **the above circuit is known as a *voltage follower***. Like its one-transistor counterpart, the common-collector ("emitter-follower") amplifier, it has a voltage gain of 1, a high input impedance, a low output impedance, and a high current gain.
- **Voltage followers are also known as *voltage buffers***, and are used to **boost the current-sourcing ability of voltage signals too weak** (too high of source impedance) to directly drive a load.
- **The op-amp model shown in the last illustration** depicts how the **output voltage** is essentially **isolated from the input voltage**, so that **current on the output pin is not supplied by the input voltage source** at all, but rather **from the power supply powering the op-amp**.

- It should be mentioned that **many op-amps cannot swing their output voltages exactly to +V/-V power supply rail voltages**.
- The model 741 is one of those that cannot: when saturated, its output voltage peaks within about one volt of the +V power supply voltage and within about 2 volts of the -V power supply voltage.
- Therefore, with a split power supply of +15/-15 volts, a 741 op-amp's output may go as high as +14 volts or as low as -13 volts (approximately), but no further.

- This is due to its bipolar transistor design. These two voltage limits are known as the *positive saturation voltage* and *negative saturation voltage*, respectively.
- Other op-amps, such as the model 3130 with field-effect transistors in the final output stage, have the ability to swing their output voltages within millivolts of either power supply *rail* voltage.
- Consequently, their positive and negative saturation voltages are practically equal to the supply voltages.

To Sum Up:
- Connecting the output of an op-amp to its inverting (-) input is called *negative feedback*. This term can be broadly applied to any dynamic system where the output signal is "fed back" to the input somehow so as to reach a point of equilibrium (balance).

- When the output of an op-amp is *directly* connected to its inverting (-) input, a *voltage follower* will be created. Whatever signal voltage is impressed upon the noninverting (+) input will be seen on the output.

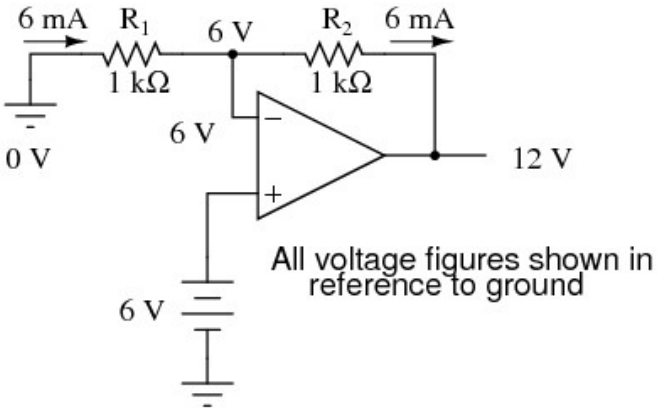
- An op-amp with negative feedback will try to drive its output voltage to whatever level necessary so that the differential voltage between the two inputs is practically zero. The higher the op-amp differential gain, the closer that differential voltage will be to zero.

- Some op-amps cannot produce an output voltage equal to their supply voltage when saturated. The model 741 is one of these. The upper and lower limits of an op-amp's output voltage swing are known as *positive saturation voltage* and *negative saturation voltage*, respectively.

Divided feedback

- If we **add a voltage divider to the negative feedback wiring** so that **only a *fraction* of the output voltage is fed back to the inverting input instead of the full amount**, the **output voltage will be a *multiple* of the input voltage** (please bear in mind that the power supply connections to the op-amp have been omitted once again for simplicity's sake):

The effects of divided negative feedback

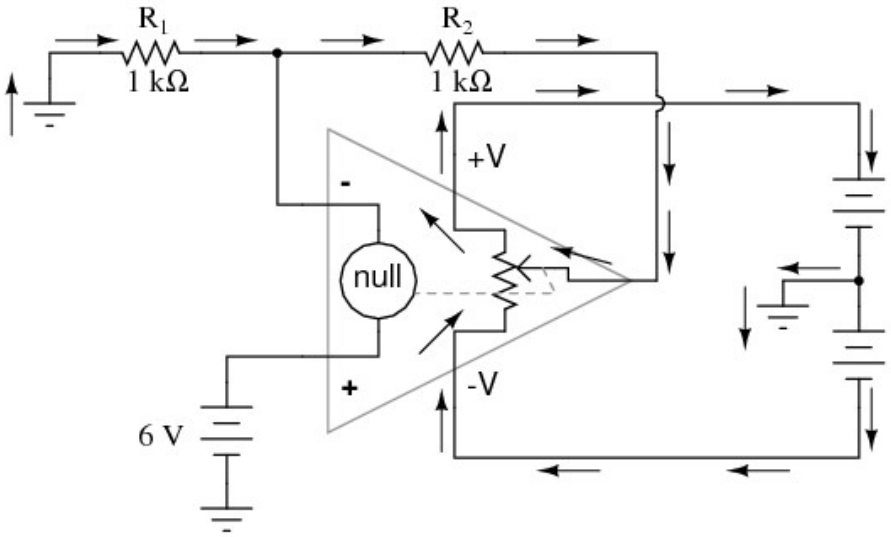


- **If** R_1 and R_2 are both equal and **V_{in} is 6 volts**, **the op-amp will output whatever voltage is needed to drop 6 volts** across R_1 (**to make the inverting input voltage equal to 6 volts, as well**, keeping the voltage difference between the two inputs equal to zero).
- **With the 2:1 voltage divider** of R_1 and R_2, **this will take 12 volts at the output** of the op-amp to accomplish.
- Another way of analyzing this circuit is to start by calculating the magnitude and direction of current through R_1, knowing the voltage on either side (and therefore, by subtraction, the voltage across R_1), and R_1's resistance.
- Since the left-hand side of R_1 is connected to ground (0 volts) and the right-hand side is at a potential of 6 volts (due to the negative feedback holding that point equal to V_{in}), we can see that we have 6 volts across R_1.
- This gives us 6 mA of current through R_1 from left to right.

- Because we know that both inputs of the op-amp have extremely high impedance, we can safely assume they won't add or subtract any current through the divider.
- In other words, we can treat R_1 and R_2 as being in series with each other: all of the electrons flowing through R_1 must flow through R_2.
- Knowing the current through R_2 and the resistance of R_2, we can calculate the voltage across R_2 (6 volts), and its polarity.
- Counting up voltages from ground (0 volts) to the right-hand side of R_2, we arrive at 12 volts on the output.

- Upon examining the last illustration, one might wonder, "**where does that 6 mA of current go**?"
- The last illustration doesn't show the entire current path, but **in reality it comes from the negative side of the DC power supply**, through ground, through R_1, through R_2, through the output pin of

the op-amp, and then back to the positive side of the DC power supply through the output transistor(s) of the op-amp.
- Using the null detector/potentiometer model of the op-amp, the current path looks like this:



- **The 6 volt signal source does not have to supply any current for the circuit**: it merely commands the op-amp to balance voltage between the inverting (-) and noninverting (+) input pins, and in so doing produce an output voltage that is twice the input due to the dividing effect of the two 1 kΩ resistors.
- **The voltage gain of this circuit**, overall, **can be changed by just by adjusting the values of R_1 and R_2** (changing the ratio of output voltage that is fed back to the inverting input).
- **Gain can be calculated by the following formula**:

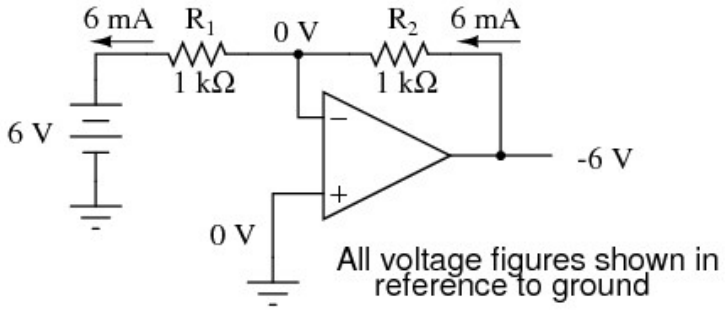
$$A_V = \frac{R_2}{R_1} + 1$$

- **Note** that **the voltage gain for this design of amplifier circuit can never be less than 1**.
- If we were to lower R_2 to a value of zero ohms, our circuit would be essentially identical to the voltage follower, with the output directly connected to the inverting input.
- Since the voltage follower has a gain of 1, this sets the lower gain limit of the noninverting amplifier.
- **However, the gain can be increased far beyond 1, by increasing R_2 in proportion to R_1**.

- **Also note that the polarity of the output matches that of the input, just as with a voltage follower**.
- A **positive input voltage results in a positive output** voltage, **and vice versa** (with respect to ground).
- For this reason, **this circuit is referred to as a *noninverting amplifier*.**

- Just as with the voltage follower, we see that the differential gain of the op-amp is irrelevant, so long as its very high.
- The voltages and currents in this circuit would hardly change at all if the op-amp's voltage gain were 250,000 instead of 200,000.
- This stands as a stark contrast to single-transistor amplifier circuit designs, where the Beta of the individual transistor greatly influenced the overall gains of the amplifier.
- With negative feedback, we have a self-correcting system that amplifies voltage according to the ratios set by the feedback resistors, not the gains internal to the op-amp.

- Let's see what happens if we retain negative feedback through a voltage divider, but apply the input voltage at a different location:



- By grounding the noninverting input, the negative feedback from the output seeks to hold the inverting input's voltage at 0 volts, as well.
- For this reason, the inverting input is referred to in this circuit as a ***virtual ground***, being held at ground potential (0 volts) by the feedback, yet not directly connected to (electrically common with) ground.
- The input voltage this time is applied to the left-hand end of the voltage divider (R_1 = R_2 = 1 kΩ again), so the output voltage must swing to -6 volts in order to balance the middle at ground potential (0 volts).
- Using the same techniques as with the noninverting amplifier, we can analyze this circuit's operation by determining current magnitudes and directions, starting with R_1, and continuing on to determining the output voltage.

- We can change the overall voltage gain of this circuit, overall, just by adjusting the values of R_1 and R_2 (changing the ratio of output voltage that is fed back to the inverting input). **Gain can be calculated by the following formula:**

$$A_V = \frac{-R_2}{R_1}$$

- **Note that this circuit's voltage gain *can* be less than 1**, depending solely on the ratio of R_2 to R_1.
- Also **note that the output voltage is always the opposite polarity of the input voltage**.
- A positive input voltage results in a negative output voltage, and vice versa (with respect to ground).
- For this reason, this circuit is referred to as an ***inverting amplifier***.
- Sometimes, the gain formula contains a negative sign (before the R_2/R_1 fraction) to reflect this reversal of polarities.

- **These two amplifier circuits** we've just investigated serve the purpose of **multiplying or dividing the magnitude of the input voltage signal**.
- **This is** exactly **how** the mathematical operations of **multiplication and division are typically handled in analog computer circuitry**.

To Sum Up:

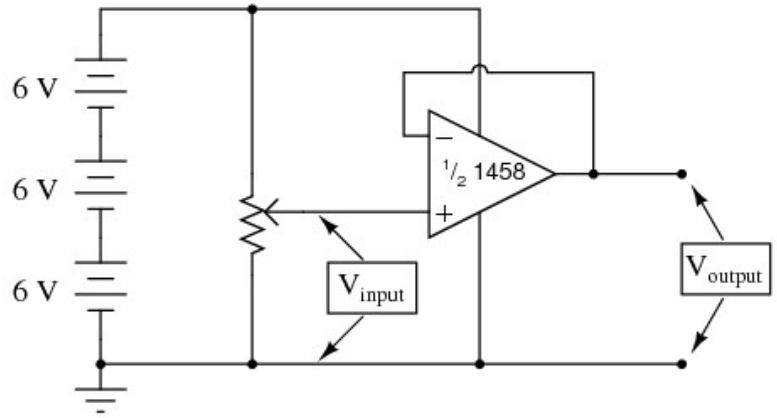
- By connecting the inverting (-) input of an op-amp directly to the output, we get **negative feedback**, which gives us a ***voltage follower*** circuit. By connecting that negative feedback through a **resistive voltage divider** (feeding back a *fraction* of the output voltage to the inverting input), the **output voltage becomes a *multiple* of the input voltage**.

- A **negative-feedback** op-amp circuit with the input signal going to the noninverting (+) input is called a ***noninverting amplifier***. The output voltage will be the same polarity as the input. **Voltage gain** is given by the following equation: $A_V = (R_2/R_1) + 1$

- A negative-feedback op-amp circuit with the **input signal going to the "bottom" of the resistive voltage divider, with the noninverting (+) input grounded, is called an *inverting amplifier***. Its **output voltage will be** the **opposite polarity** of the input. **Voltage gain is** given by the following equation: $A_V = -R_2/R_1$

Precision voltage follower

SCHEMATIC DIAGRAM



INSTRUCTIONS

- if we want the op-amp to behave as a true *amplifier*, we need it to exhibit a manageable voltage gain.
- Since we do not have the luxury of disassembling the integrated circuitry of the op-amp and changing resistor values to give a lesser voltage gain, we are limited to external connections and componentry.
- Actually, this is not a disadvantage as one might think, because the combination of extremely high open-loop voltage gain coupled with feedback allows us to use the op-amp for a much wider variety of purposes, much easier than if we were to exercise the option of modifying its internal circuitry.

- If we connect the output of an op-amp to its inverting (-) input, the output voltage will seek whatever level is necessary to balance the inverting input's voltage with that applied to the noninverting (+) input.
- If this feedback connection is direct, as in a straight piece of wire, the output voltage will precisely "follow" the noninverting input's voltage.
- Unlike the *voltage follower* circuit made from a single transistor (see chapter 5: Discrete Semiconductor Circuits), which approximated the input voltage to within several tenths of a volt, this voltage follower circuit will output a voltage accurate to within mere *microvolts* of the input voltage!

- Many op-amps, the specified models included, cannot "swing" their output voltage exactly to full power supply ("rail") voltage levels.
- In this case, the "rail" voltages are +18 volts and 0 volts, respectively.
- Due to limitations in the 1458's internal circuitry, its output voltage is unable to exactly reach these high and low limits.
- You may find that it can only go within a volt or two of the power supply "rails."
- This is a very important limitation to understand when designing circuits using operational amplifiers.

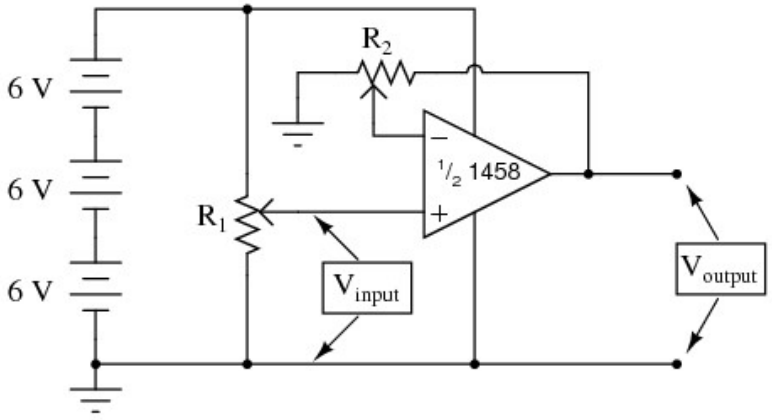
- If full "rail-to-rail" output voltage swing is required in a circuit design, other op-amp models may be selected which offer this capability.
- The model 3130 is one such op-amp.

- Precision voltage follower circuits are useful if the voltage signal to be amplified cannot tolerate "loading;" that is, if it has a high source impedance.
- Since a voltage follower by definition has a voltage gain of 1, its purpose has nothing to do with amplifying voltage, but rather with amplifying a signal's capacity to deliver *current* to a load.

- Voltage follower circuits have another important use for circuit builders: they allow for simple linear testing of an op-amp.
- One of the troubleshooting techniques recommended is to *simplify and rebuild*. Suppose that you are building a circuit using one or more op-amps to perform some advanced function.
- If one of those op-amps seems to be causing a problem and you suspect it may be faulty, try re-connecting it as a simple voltage follower and see if it functions in that capacity.
- An op-amp that fails to work as a voltage follower certainly won't work as anything more complex!

Noninverting amplifier

SCHEMATIC DIAGRAM



INSTRUCTIONS
- This circuit differs from the voltage follower in only one respect: output voltage is "fed back" to the inverting (-) input through a voltage-dividing potentiometer rather than being directly connected.
- With only a *fraction* of the output voltage fed back to the inverting input, the op-amp will output a corresponding *multiple* of the voltage sensed at the noninverting (+) input in keeping the input differential voltage near zero.
- In other words, the op-amp will now function as an amplifier with a controllable voltage gain, that gain being established by the position of the feedback potentiometer (R_2).

- Because the output voltage increases in a positive direction for a positive increase of the input voltage, this amplifier is referred to as *noninverting*.
- If the output and input voltages were related to one another in an inverse fashion (i.e. positive increasing input voltage results in positive decreasing or negative increasing output), then the amplifier would be known as an *inverting* type.
- The ability to leverage an op-amp in this fashion to create an amplifier with controllable voltage gain makes this circuit an extremely useful one.
- It would take quite a bit more design and troubleshooting effort to produce a similar circuit using discrete transistors.

This knowledge of Diodes and OpAmps gives us the tools to construct near every form of analogue logic circuit that can be built:

Analogue Logic Funtions and How They Are Achieved:

- **CV Mixer** }
- **Audio mixer** } - uses OpAmps to perform **summing/addition**
- **Matrix mixer** }
- **polarising mixers** – use OpAmps to further perform **subtraction** also.

_____

- **VCA** – uses operational transconductance amplifiers or transistors configured as VC variable resistors to perform **multiplication and squaring**
- **Polarising VCAs/4 Qdnt. Multipliers/Ring Mods** – use the same circuitry as VCAs to perform *multiplication, squaring (freq doubling), division, square root, phase angle detection and rectification and much more besides:*

Some Applications of Analogue Multipliers:

analogue multipliers can be used, amongst a wider range of things,for the following purposes:
- Voltage Squarer
- Frequency doublers
- Voltage divider
- Square rooter
- Phase angle detector
- Rectifier

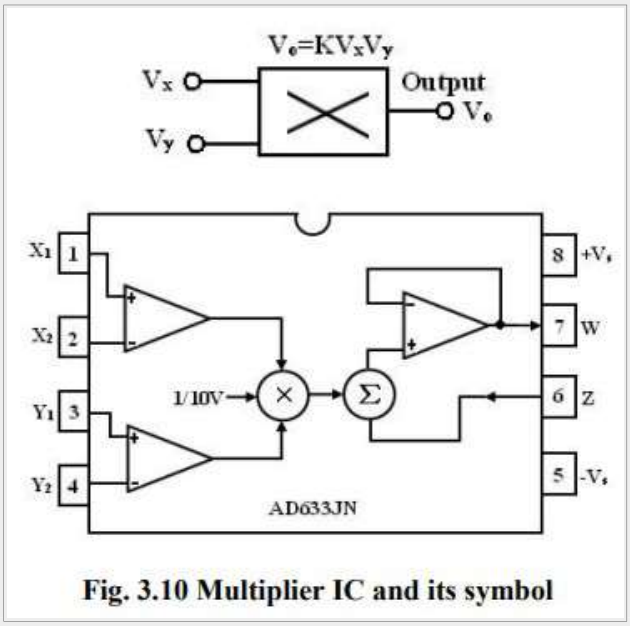


Fig. 3.10 Multiplier IC and its symbol

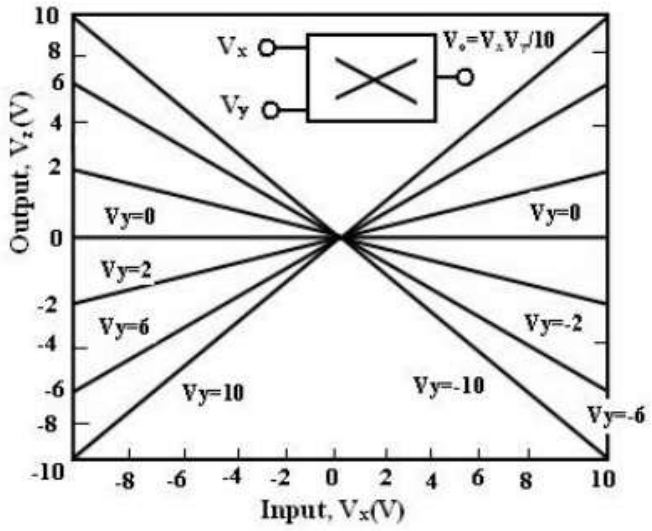


Fig.3.11 Transfer characteristics of a typical four-quadrant multiplier

Voltage Squarer:

- The figure shows the multiplier connected as a squaring circuit.
- The inputs can be positive or negative, represented by any corresponding voltage level between 0 and 10V.
- The input voltage Vi to be squared is simply connected to both the input terminals, and hence we have,

Vx = Vy = Vi

and the output is

V0 = Kvi2

- The circuit thus performs the squaring operation.
- This application can be extended for frequency doubling applications.

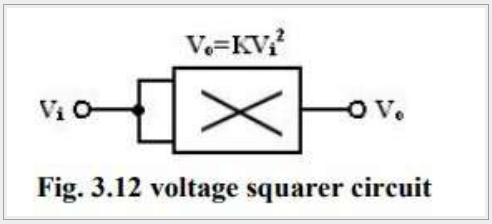


Fig. 3.12 voltage squarer circuit

Frequency doublers:

- The figure shows the squaring circuit connected for frequency doubling operation.
- A sine-wave signal Vi has a peak amplitude of Av and frequency of f Hz.
- Then, the output voltage of the doublers circuit is given by

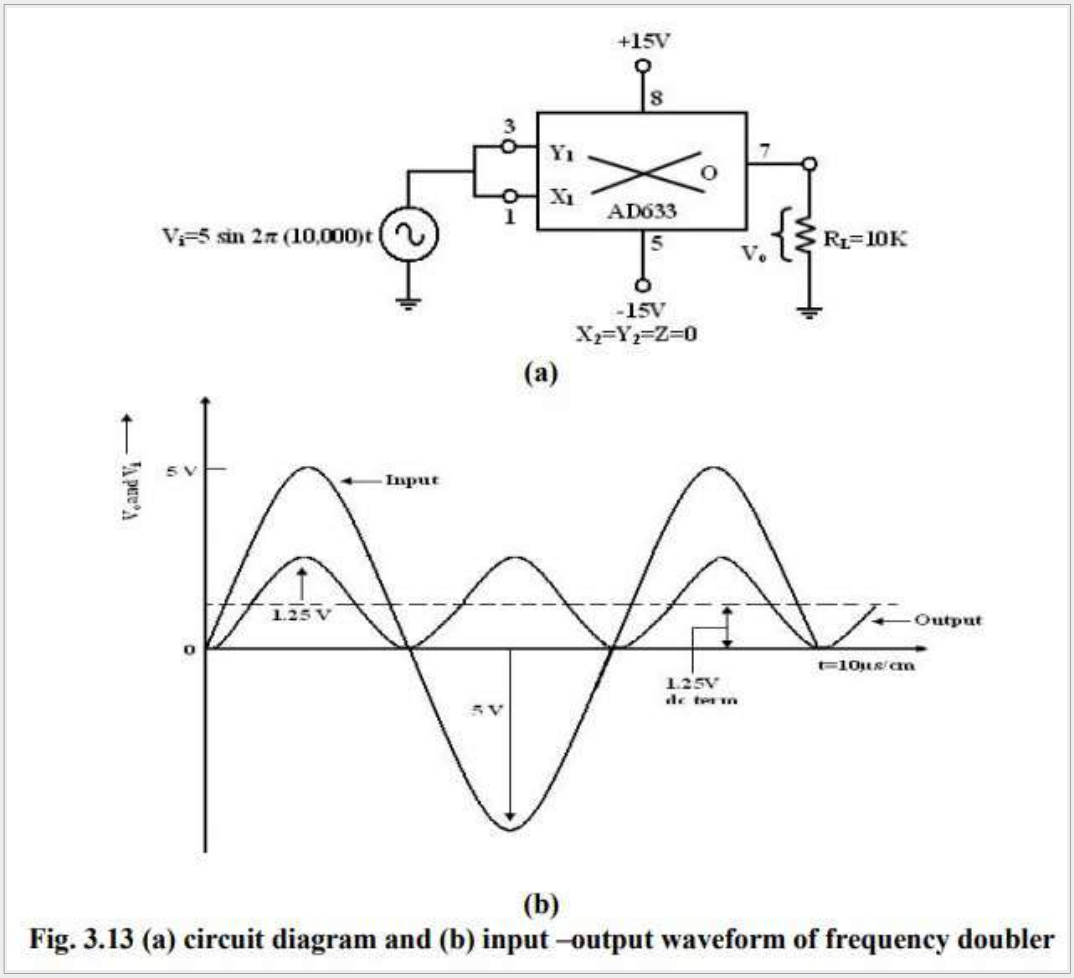


Fig. 3.13 (a) circuit diagram and (b) input –output waveform of frequency doubler

- Assuming a peak amplitude Av of 5V and frequency f of 10KHz,

*V0 = 1.25 – (1.25 x cos2 x 20000)t ***CHECK****

- The first term represents the dc term of 1.25V peak amplitude.
- The input and output waveforms are shown in figure.
- The output waveforms ripple with twice the input frequency in the rectified

output of the input signal.
- This forms the principle of application of analog multiplier as rectifier of ac signals.

- The dc component of output V0 can be removed by connecting a 1μF coupling capacitor between the output terminal and a load resistor, across which the output can be observed.

Voltage Divider:

- In a voltage divider circuit the division is achieved by connecting the multiplier in the feedback loop of an op-amp.

- The voltages Vden and Vnum represent the two input voltages, Vdm forms one input of the multiplier, and output of op-amp VoA forms the second input.

- The output VOA forms the second input.
- The output VOM of the multiplier is connected back of op-amp in the feedback loop.
- Then the characteristic operation of the multiplier gives:

Vom = KVOA Vdm (1)

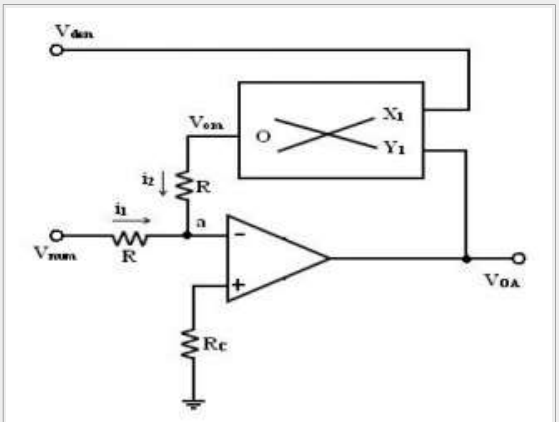


Fig 3.14 divider circuit

- As shown in the figure, no input signal current can flow into the inverting input terminal of op-amp, which is at virtual ground.
- Therefore, at the junction a,

i1 + i2 =0,
the current i1 = Vnum / R
where R is the input resistance and,
the current i2 = Vom /R

With virtual ground existing at a,

i1+i2 = Vnum / R + Vom /R = 0

KVOA Vden = - Vnum

or

voA=- vnum/Kvden

where Vnum and Vden are the numerator and denominator voltages respectively.

- Therefore, the voltage division operation is achieved.
- Vnum can be a positive or negative voltage and Vden can have only positive values to ensure negative feedback.
- When Vdm is changed, the gain 10/Vdm changes, and this feature is used in automatic gain control (AGC) circuits.

Square Rooter:

- The divider voltage can be used to find the square root of a signal by connecting both inputs of the multiplier to the output of the op-amp.
- Substituting equal in magnitude but opposite in polarity (with respect to ground) to Vi.
- But we know that Vom is one-term (Scale factor) of

*V0 * V0 or -Vi = Vom = V2/1 0*

Solving for V0 and eliminating √-1 yields,

V0 = √10|Vi |

- the Eqn. states that V0 equals the square root of 10 times the absolute magnitude of Vi.

- The input voltage Vi must be negative, or else, the op-amp saturates.
- The range of Vi is between -1 and -10V. Voltages less than -1V will cause inaccuracies in the result.

- The diode prevents negative saturation for positive polarity Vi signals.
- For positive values of Vi the diode connections are reversed.

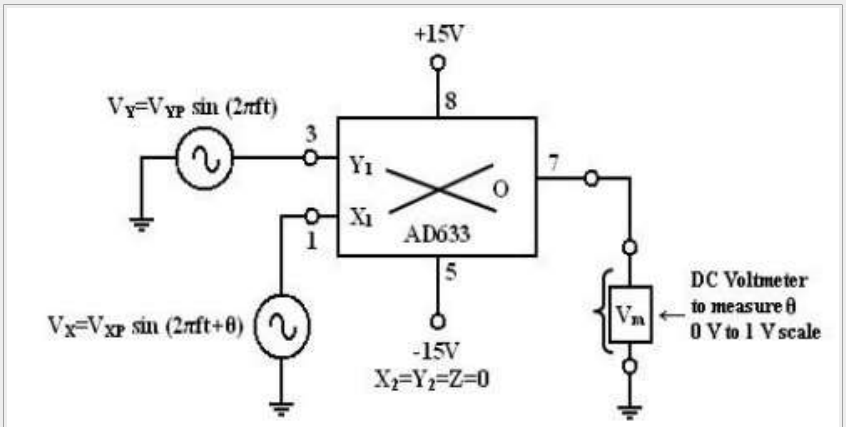


Fig 3.15 Phase angle measurement circuit diagram

Phase Angle detector:

- The multiplier configured for phase angle detection measurement is shown in the figure.
- When two sine-waves of the same frequency are applied to the inputs of the multiplier, the output V0 has a dc component and an AC component.

- The trigonometric identity shows that

Sin A sin B =1/2 (cos (A-B) – cos (A+B))

- When the two frequencies are equal, but with different phase angles, e.g. A = 2πft +θ for signal Vx and,

B = 2πft for signal Vy, then using the identity

[sin (2 ft+)][sin2 ft)]
= 1/2[cos -cos(4 ft +)]
= 1/2(dc- the double frequency term)

- Therefore, when the two input signals Vx and Vy are applied to the multiplier, V0 (dc) is given by:

$$v_v(dc) = \frac{v_{xp}v_{yp}}{20}\cos\theta$$

- where Vxp and Vyp are the peak voltage amplitudes of the signals Vx and Vy.
- Thus, the output V0(dc) depends on the factor cos θ. A dc voltmeter can be calibrated as a phase angle meter when the product of Vxp and Vyp is made equal to 20.
- Then, a (0-1) V range dc voltmeter can directly read cos θ, with the meter calibrated directly in degrees from a cosine table.
- The input and output waveforms are shown in the figure.

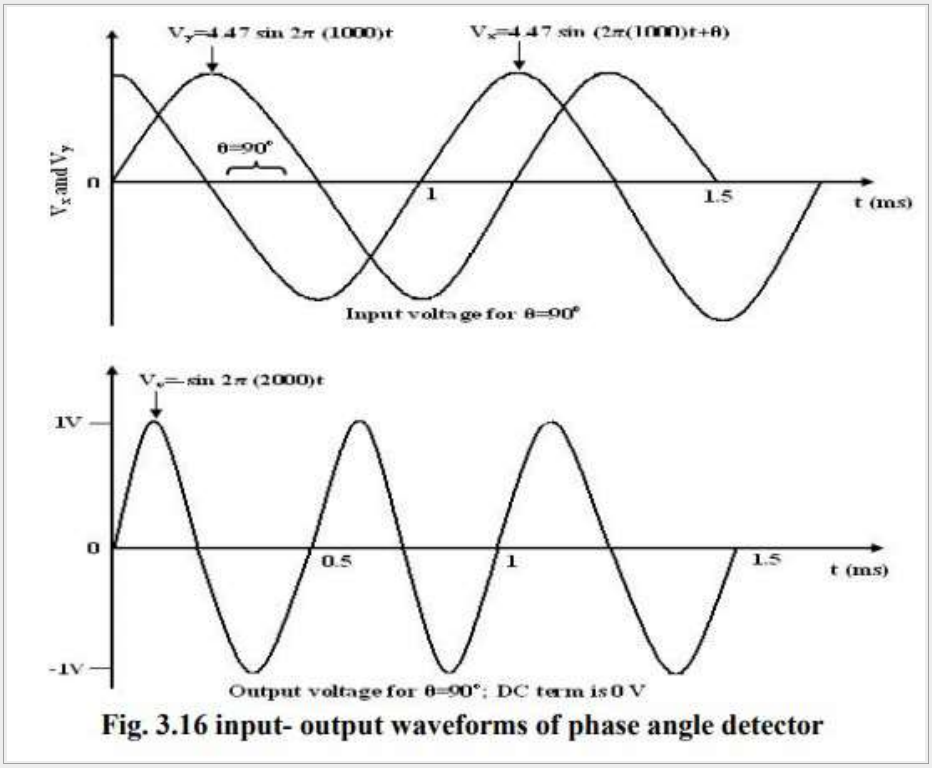


Fig. 3.16 input- output waveforms of phase angle detector

- Then the above eqn becomes:

V0 (dc) = cos θ

if we make the product Vxp Vyp = 20
or, in other words,

Vxp – Vyp = 4.47V.

- semi analogue AND – **MIN()** functions achieved via **diode AND** circuits perform certain analogue functions but not the full range a VCA does.

- semi analogue OR –**MAX()** functions achieved via **diode OR** circuits can perform certain analogue functions but not the full range an opamp mixer can.

- [only the non-inverting logical AND and logical OR functions can be realized by diode gates].

- Analogue **SORT** can be configured to do anything from put two values low to high/vice versa, to ordering a whole list.
- second largest or second smallest of three, when using a 3 input diode logic circuit gives **analogue median**.
- these are actually special cases of applied MAX() and MIN()

combinations.

_____

- crossfaders
VC crossfaders can kinda work as logic modules:

- **AND GATE** aka 'MIN' = Signal A -> Input 1 / Signal B -> CV (Basically this is same as a linear VCA)
- **OR GATE** aka 'MAX' = Signal A -> Input 1 & CV / Signal B -> Input 2
- **XOR GATE** aka 'Ring Mod' = Signal A -> Input 1 / Inverted Signal A -> Input 2, Signal B -> CV Input

note, however, AND & OR are limited to gate [i.e. digital] only outputs, whereas, the **XOR is true analogue** with all CV signals.

_____

- **bi-directional switches**
- whilst not technically true analogue logic, they give interesting results when processing CVs and sometimes even audio:
- **momentary** mode produces (if 1 input to 2 output) **IF, THEN, ELSE CONDITIONAL LOGIC** functions when triggered (>certain voltage threshold), then output 2, else output 1.

- **momentary** mode produces (when 2 input to 1 output) a **NOT GATE** - if the gate is high by default, gate is low when triggered.

- **latch** mode produces (if 1 input to 2 output) **FLIP-FLOP LOGIC** when triggered - flip-flops output 1 & output 2 alternate.

_____

- **Diode inverters** make positive input only **NOT** gates

- **OpAmp inverters** make an true analogue inversion – i.e. a **NOT** function.

_____

- **attenuation** – even the most basic manual attenuator, a simple passive potentiometer or even static pad type level reducer is performing a **division** function.

_____

- **attenuversion** – see above – **signed subtraction**?

_____

- **offset** – static offset voltage sources provide the function of a mathematical **constant** function in a given calculation or circuit.

_____
- **slope detector & comparators** – perform **dependent logic** functions although their outputs are digital not analogue.

- with only **three comparators**, **any possible boolean gate** can be

constructed.

- **Analogue 'comparator' circuits** can be constructed that **output** a continuous voltage **between 0 and 100%** dependent on a predefined setting to be triggered by an **OpAmp threshold comparator** combined with a **DC offset** source.
- this case of an analogue comparator circuit can be applied to **dynamic comparator** functions, **window comparators** and hence **combinations** thereof.

_____

- **sequential switches** and **matrices** – a variety of **complex** logic functions such as multi-operation math functions similar to the **IF, THEN, ELSE and FLIPFLOP** logic achievable by momentary and latching bi-directional switches.

_____

- **MAJORITY** logic built from diodes can give an analogue **'(a and b) or (a and c) or (b and c)'** function.

_____

- **ODD/EVEN PARITY** gates constructed from diodes - XOR into XOR gives **'(A o+ B) o+ C'** functions.

_____

- **IMPLY** gates built from diodes – [result in **+ve rectification**?]
[3 way rectify module?]
- **NIMPLY** gates from diode logic – [result in **-ve rectification??**]

_____

SOME IDEAS FOR FINDING ANALOGUE LOGIC IN EURORACK:

Some readily available analogue logic modules in the eurorack world can be listed off easily, but not all are apparent to everyone as having analogue logic possibilities.

The usual candidate for Analog Logic Modules that spring to mind:
- **Mannequins** - Cold Mac
- **Mystic Circuits** - ANA
- **Mutable Instruments** - Kinks
- **WMD** - OSD
- **Intellijel** - µMod II
- **Ladik** – Median,
- Any MIN/MAX – there's so many out there I don't need to list em!
- **Intellijel** - Bifold
-

Less obvious ones you may now want to think about:

VC crossfaders e.g.
- **Happy Nerding** – Xfade
- **WMD** – AXYS
- **RYO/Kymatica Devices** - 2xVCX
-

Bi-directional momentary switches and switches with latches e.g.
- **Instruo** – Tain
- **Ritual Electronics** - Pointeuse
- **RYO** – DT Switch
- **RYO** - Paths

also consider:
- VC slope gens, slew limiters and other DUSG type modules such as **Makenoise** Maths or the **Doepfer** A-171-2
- slope detectors
- envelope followers
- s&h/t&h
- envelopes and contour generators
- unusual mixers
- static offset voltage sources
-

in other words many basic utilities and everyday 'plain basic building block' type modules prove to be the best analogue logic tools, and always remember: feedback can do wonderful things!

.....

The following math derivations and calculations are purely for the intellectual interest of those who have survived this far – they are in no way essential to understand anything else on this site or any of the patch ideas, module modifications or anything else I write about:

$\in$ Denotes set membership, and is read "in" or "**belongs to**". That is, $x \in X$ means that x is an element of the set X.

∪ Denotes the union of two sets A and B is the set of elements which are **in A, in B, or in both A and B**. In symbols, A ∪ B = {x:x ∈ A or x ∈ B}.

∩ Denotes the intersection of two sets A and B is the set of elements which are **in both A and B**. In symbols, A ∩ B = {x:x ∈ A and x ∈ B}

∨ Denotes the inclusive disjunction of two sets A and B. Is true if **either A or B, or both A and B** are true.

∧ Denotes the conjunction of two sets A and B. Is true if **both A and B** are true.

Now to use those symbols to aid us in mathematically demonstrating the workings of analogue logic:

NOT/Inverting:
- Complement. The complement groups all the elements that do not reside in the set μ(x).

$$\mu(x) = 1 - \mu(x),\ x \in X$$

.....

Gain:
- mathematically the Scalar product. An analogue set can be multiplied by a scalar S.

$$\mu(x) = S \cdot \mu_1(x),\ x \in X$$

.....

Square: [special case*]
- mathematically raising to a Power. The power operation elevates an analogue set to a particular number m.

$$\mu(x)=[\mu_1(x)]m,\ x \in X$$

*The case m = 2 is known as the concentration of a fuzzy set. 'The concentration is the result of putting the same value into both inputs of a bipolar VCA or four quadrant multiplier.

.....

Max()/Multiply: [diode MAX, VCA, VC X-fade, 4-Quadrant Multiplier/Bi-polar VCA/Ring Mod.]

- mathematically the Union. The union of two or more analogue sets joins all the elements of the universe of discourse that belong in some degree to any of those sets. This operation can be done with any triangular co-norm. In this particular implementation, we unite fuzzy sets by selecting the maximum values among them.

$$\mu\cup(x) = \mu_1(x) \vee \mu_2(x) \vee \ldots \vee \mu n(x) = \max(\mu_1(x), \mu_2(x), \ldots, \mu n(x)),\ x \in X$$

.....

Addition: [MIN, mixer/varieties of special mixer]
- mathematically the Intersection. The intersection of two or more analogue sets extracts all the elements of the universe of discourse that belong in some degree to all of those sets. This operation can be done with any triangular norm. In this particular implementation, we unite fuzzy sets by selecting the minimum values among them.

$$\mu\cap(x) = \mu_1(x) \wedge \mu_2(x) \wedge \ldots \wedge \mu n(x) = \min(\mu_1(x), \mu_2(x), \ldots, \mu n(x)),\ x \in X$$

.....

From the above we can derive the math for, for example, a **4x4 matrix mixer**:

Any given output is:

$$\mu\cap(x_i y_o) = \mu_1(xiyo) \wedge \mu 2\,(x_i y_o) \wedge \ldots \wedge \mu n\,(x_i y_o)$$
$$= \min(\mu_1(x_i y_o), \mu_2(x_i y_o), \mu_3(x_i y_o), \mu_4(x_i y_o)),\ x_i y_o \in X_i Y_o$$

Where x_i is the horizontal row number, and y_i the vertical column number

Where a row x_1 is:

$$\mu\cap(x_1 y_o) = \mu_1(x_1 y_1) \wedge \mu_2(x_1 y_2) \ldots \wedge \mu_n(x_1 y_o)$$
$$= \min(\mu_1(x_1 y_1), \mu_2(x_1 y_2), \mu_3(x_1 y_3), \mu_4(x_1 y_4)),\ x_1 y_o \in X_1 Y_o$$

And a column y_3 is:

$$\mu\cap(x_i y_3) = \mu_1(x_1 y_3) \wedge \mu_2(x_2 y_3) \wedge \ldots \wedge \mu_n(x_i y_3)$$
$$= \min(\mu_1(x_1 y_3), \mu_2(x_2 y_3), \mu_3(x_3 y_3), \mu_4(x_4 y_3)),\ x_i y_3 \in X_i Y_3$$

.....

similarly, following some expanding of/breaking down into basic concepts it can be demonstrated that the relevant mathematics can be derived for all the above analogue logic tools using just these basic pieces of fuzzy logic mathematics.

.....

IF, AND, THEN

the min-max method where an analogue rule would have the form:

IF x_1 is A_1k AND x_2 is A_2k THEN yk is Bk for k=1,2,...

where A_1k and A_2k are ananlogue inputs and Bk is the desired output.

For r disjunctive analogue IF-THEN rules, the analogue output will be:

$\mu Bk(y) = \max k [\min [\mu A_1k (\text{input } (1)), \mu A_2k (\text{input } (2)), \ldots]]$ for k = 1, 2, ..., r

Looking at the XOR truth table we can write it out long form:

XOR truth table

Input		Output		
A	B	A + B	A x B	
0	0	0	1	(A AND B)
0	1	1	0	(A OR B)
1	0	1	0	(A OR B)
1	1	0	1	(A AND B)

To produce the proof:

A XOR B = 1 - (A AND B) **AND (A OR B)**
= (1 - A OR 1 - B) **AND (A OR B)**
= ([A OR 1 - B] AND [1 - A OR 1 – B]) **AND ([A OR B] AND [1 - A OR B])**
= ([A AND 1 - B] OR 1 - B) **AND ([A AND 1 - B] OR B)**

Therefore

A XOR B = (A AND [1 - B]) OR ((1 - A) AND B)

Using the above, we can deduce the mathematical formula for :

IF, THEN, ELSE

IF x_1 is A_1k THEN y_1 is Bk ELSE z_1 is Ck for k=1, 2, ...

Where A_1k is an analogue input and Bk and Ck are outputs.

For r disjunctive analogue IF-THEN rules, the analogue output will be:

uBk(y) = XOR [μA_1k, μA_2k, ...] for k = 1, 2 ,..., r

uBk(y) = [1 - μA_1k AND μA_2k] OR [1 - ... AND ...] for k = 1, 2, ..., r

μBk(y) = min [max [1 – μA_1k (input 1), μA_2k (input 2)], max [1 - ..., ...]] for k = 1, 2, ..., r

Isn't that just great :)

[yeah, ok, enough – lets go build an analogue computer from serge!!!]

.....

- As you may have noticed earlier on there was a brief mention of a logic gate i've yet to cover – the '**imply**' gate and it's inverted counterpart, the '**nimply**' gate so here's an intro to those two guys:

IMPLY Gate

The **IMPLY** logic gate implements a “logical conditional”. It forms the statement “**If A then B**”.

- The output of this gate is, “if the input A is true then input B is also true” So when A is True “HIGH” and B is True “HIGH”, the output is true “HIGH”.

- When A is true but B is false then the output is false.

- Now if the input A is false then automatically the output becomes true regardless of input B i.e. the output is True in both cases.

A	B	Output
0	0	1
0	1	1
1	0	0
1	1	1

NIMPLY Gate

- **NIMPLY** Logic gate has an inverted output of the IMPLY Logic gate.

- It implements the statement "**If A but not B**".

- It means the output is true when and only when the input A is true but B is False.

A	B	Output
0	0	0
0	1	0
1	0	1
1	1	0

- Note these gates can be built in analogue circuitry using opamps and other true analogue gate components but will not work as diode logic due to the nature of diode NOT gates/Inverters.
- As a result of the above, to achieve the analogue proofs and concept demonstrations in the following set of examples, diode logic can't be considered, only actual OpAmp inverters and mixers.
- Although technically diode OR fucnctions could be used, a mix of diode and OpAmp logic being workable, if you're already using Opamp inverters, you got a quad OpAmp IC on the go most likely – why not use the remaining amplifiers to do the mixing...

- the imply gate works out as a simple subtraction device and output is equal to the difference between the two inputs.
- since diode logic doesn't work consistently for inversion, an imply gate works somewhat weirdly in analogue logic situations;

- as a result, it’s definitely worth noting that the inputs on these are taken to be magnitude only so you can’t use negative numbers within a subtraction and achieve the correct arithmetic result.
- The output can be negative and will be arithmetically correct, assuming that both inputs are positive. This might be a bit confusing, so here are some examples:

Inputs +	Inputs -	Correct Answer	imply Output
100	35	65	65
50	65	-15	-15
50	-10	60	40
-50	30	-80	20

- When including negative values in the subtraction, as either operand, the result is incorrect.

50 – (-10) should give 60

however the actual calculation performed is:

50 – 10, to give 40.

- As this is generally only going to be used in patching up fun cv and audio signal manipulation creations there isn't really a problem with this behaviour, it juts results in interesting quirks like wavefolding or dead spots etc., but,
- when trying to build an actual analogue computer or perform specific accurate calculations its important to bear this in mind and so OpAmp mixers should be used not diode based imply gates.

- So, now i'll explain why imply/nimply gates are suddenly of such great interest – in a very long winded way:

Generic Signal Notation:

- Generic signals will be described as follows (specific diagrams may use other labels for special cases):

Inputs: x_n
Outputs: y_n

Internals: z_n

- For example, the function of an AND gate on N analogue signals is described as:

$z = \min (x_1, x_2, x_3, ..., x_N)$

Signed and Unsigned Numbers:

- Since in analogue logic some analogue signals can take negative values and some can't (see section on diodes and diode logic), for differentiation between the two, the terms "signed", for values that can take negative values, and "unsigned" for those that can't will be used.
- [remember there's a mathematical limit of 100 for the purposes of these calculations, although it is in fact 100% of the rail voltage irl]:

Unsigned: $0 \leq s \leq 100$
Signed: $-100 \leq s \leq 100$

- Note that many of these circuits systems will only work with unsigned values. However, you will be able to use them with inputs that are technically signed as long as you can guarantee that the input values will never drop below 0.
- Alternatively you can choose to ignore the value of the signal if it is negative (forcing it to 0) but using the positive component.

Positive and Negative Components:

- Splitting signed signals is a big part of how the limitations of calculations taking absolute values only are circumvented, obtaining the positive and negative components.
- These are the positive and negative outputs of a 'splitter' subcircuit and will be annotated as follows:

Positive output: z^+
Negative output: z^-

- Note that in these cases, only one of z^+ and z^- can ever be non-zero at the same time and in both cases the signal is unsigned. This splitting is therefore very similar to taking the absolute value, in the sense that one of the two outputs of the splitter will be the absolute value and the other will be 0, so the following is true:

$\max (z^+, z^-) = |z|$

|z| is the absolute value of signal “z” :

Where :
|z| = |-z|
0 ≤ |z|

- All arithmetic calculations in the next set of examples will be using a theoretical maximum value of 100.
- this is an absolute mathematical value to make explanations easier and the math easier to understand.
- In actuality the maximum value of any equation involving voltages in analogue logic circuits would be +/-100% of rail voltage, so for eurorack modular that would generally be +/-12V.

---

so now i've defined the generic terms terms we'll be using lets see why imply and nimply gates are so useful in analogue logic:

- Given that:

z = x + y = 100 - (100 - (x + y)) is true,

- [remember 100 is the arbitrary maximum value to ease the math but irl would be 100% of rail voltage]

- And, one can add numbers together using nothing but subtraction;
- [I’ll demonstrate: rather than adding both numbers together, lets subtract them both from 100, then subtract the answer from 100]
- using values of 32 and 43 as examples, normally the calculation would be:

$z = x_1 + x_2$
z = 32 + 43
z = 75

- Using the equation above:

$z = 100 - (100 - (x_1 + x_2))$
z = 100 - (100 - (32 + 43))
z = 100 - (100 - 75)

z = 100 - (25)
z = 75

- This *looks* ridiculously long-winded when written down like that, but, there is method in my madness;

- if you remember that the above equation;

z = 100 - (100 - (x + y))

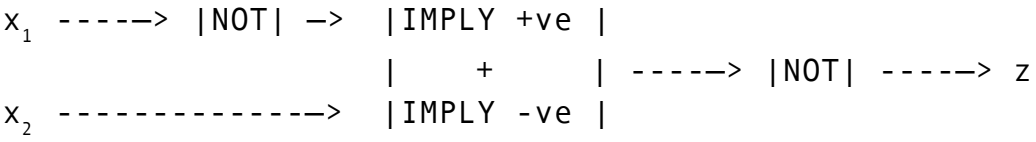
- Can be rewritten as:

z = 100 - ((100 - x) + (- y))

And,
- NOT gates perform the analogue function (100 - |x|), and,
- IMPLY gates perform a special case of signed addition, Since OR gates perform addition – imply gates perform a type of subtraction basically.

- note here that there is therefore a degree of overlap between imply gates and OpAmps in inverting amplifier configuration.

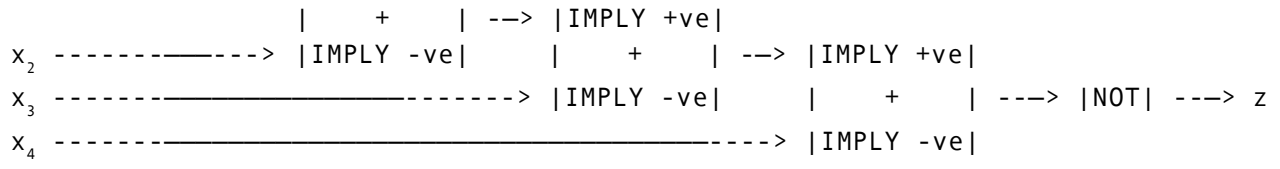
one can simplify this to give the following circuit for basic addition with 2 inputs:



do note here:
This circuit will return the arithmetically correct answer for the sum of its inputs if the following conditions are met:
- All inputs are greater than zero.
- the sum of the inputs is smaller than, or equal to 100.

- The first assumption is forced by the nature of subtraction using diode logic, i.e. it just can't handle subtraction of negatives.
- the second assumption is due to the fact that values over 100 [irl 100% of rail voltage] are out of range.

- Still long winded and could have just been done with just OR gates? Look what happens when we extend the number of inputs:



- i.e. it can then be extended for any number of inputs, and the inversions only need to be carried out at the beginning and end of the calculation - everything in between is IMPLY gates alone - simple subtraction.

- This is relevant because a [inverting OpAmp] mixer is just the inverted top input, the inverted input of each other input into summers [or ORs, or IMPLY gates] with a final inverter at the end - i.e. each stage of NOT or IMPLY can be replaced by an inverting OpAmp stage so only a quad, dual and quad or pair of quad amps will usually suffice for a whole mixer!

- Even better, you can extend this demonstration of logic into math to show the same runs true for addition and subtraction together [attenuverting mixers] or even switched non/attenuverting and [scaleable (i.e. with gain on the non-inverting) mixers - often making there a possibility of multiplying certain factors].
- Also consider static dc offsets in the mix as an option when nothings plugged in - a normalled offset gives a constant number c in the $z = c + (x_1 + x_2)$ equation, or can be used in more complex ways if extra math is occurring!

- what's super cool is this is true for binary logic with not and imply gates – the basic 1s and 0s a computer churns through and we use for trigs, gates, clocks, etc, but its as above totally analogue logic capable too - so one can use mixers as adders and hence also OR gates as interchangeably as needed,
- with the caveat that we can feed in any continuously variable data source - i.e. constant varying voltages, ac, into fractions of a voltage as small as divisions will make any difference or the until the limits of the electronics is reached.

- the mixer, whether an analogue OR gate from a pair of diodes reverse biased and stuck together, a chain of inverting OpAmps, or some transistor NOT gates, OR gates and other combos of transistors and diodes (TTL), transistors/diodes/resistors (RTL) or any other logic is still mixing something and hence can perform

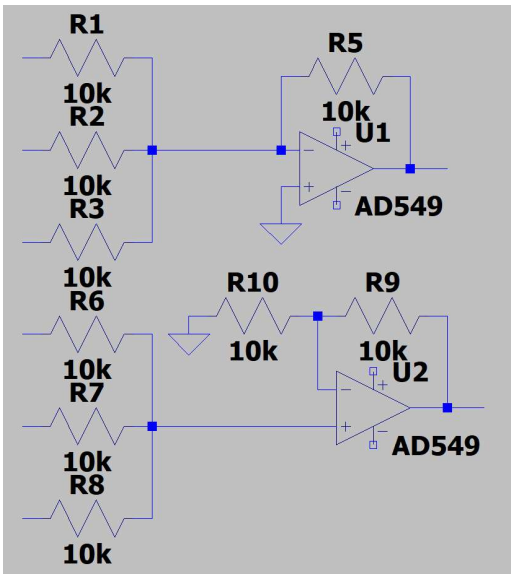
addition at bare minimum - possibly the most valuable tool in your electronic tool box for modular synthesis - they join everything else together.

.....

- Whilst writing this I was rubber ducking parts of this off friends and got asked:

“why is non-inverting buffering/mixing bad practice - what are some disadvantages of using non-inverting OpAmp configurations in a mixer?”

- the main issue is balance; in the figure below, the top mixer is balanced by the 10k over the OpAmp while the bottom one is unbalanced:



- it has to do with virtual ground, at the convergence point the node is 0v net.

- The relevance of this point is that it gets weird when you upgrade the resistors to pots - as soon as you breadboard it and start changing values with the pots, variations are created with the non-inverting and it all gets screwy in places...

- mostly the disadvantages are to do with the situations when you have one pot affecting another, things forming filters, things bleeding etc – crucially as well, inverting buffers don't need and extra stage to avoid the fact you need to first attenuate or mix or gain or whatever then add a buffer then sum and the summer will be inverting so you'll need another inverter to make the sum positive.

- its fine with just two signals, inverting gives the minimum number of steps to mix more than two.
- try it with three four or five - two signals you can do in a dual or quad OpAmp, 3, 4 or 5 is a lot pf OpAmps non-inverting.
- best way to see is breadboard it and try wiggling knobs with.a scope hooked up - feeling it out in the real world is a lot more educational than math, logic, words or simulators but a simulator will do fine for this:
- watch for things like voltage droops at all points [you don't want em happening at ins or outs - cables are relevant as well as internally], and likewise monitor amps flowing through things [it may mostly be mA, but there's still heat dissipation etc.]
- Also monitor across a whole range of frequencies, both dc, ac, moving/static etc
- look at everything and decide if you are happy(if you're lucky you accidentally connect +12v straight to ground or get OpAmp wrong way round and the power shorts, fuse or protection resistor goes bang and you get magic smoke – yay!)

- mainly i'd argue specifically inverting is better when it comes to multiple input [i.e.more than two] mixers because you need less OpAmps total to not get weird mixing artefacts with the pots
- but, as soon as you start adding attenuversion or gain etc it becomes absolutely essential because some of that simply requires a negative feedback resistor and feedback connections that are unavoidably inverting OpAmp setups

- rarely is there a call for just a simple 2 input non inverting mixer setup in a single module that is just a mixer so usually you're wanting inverting configurations

- non-inverting two input mixing is relevant sometimes in mixing together two parts of a module like a high pass and low pass feeding together for example but in reality that usually feeds perhaps an output gain stage or whatever so you end up wanting to invert the signal anyways;
- so, inverting mixers as ever usually end up used just to save

extra stages and minimise OpAmp totals.

.....

I was also quizzed on the ideal values for the input/feedback resistor pairs;

- i like 47k, but, if i'm using attenuversion in a mixer then i use 100k cos the pots can be 100k and its easy to get the pots.
- if its for some reason internal in a module and small values are being used i might use 10k occasionally but thats rare
- i like 47k because it leaves room both sides to scale down or up, its the middle number on the E series and its not huge heat dissipation anywhere or a crazy vulnerable to being off if its a 1%

”why is the standard value 47 and not 50?”

- Each E series subdivides each decade magnitude into steps of 3, 6, 12, 24, 48, 96, 192 values.
- Subdivisions of E3 to E192 ensure the maximum error will be divided in the order of 40%, 20%, 10%, 5%, 2%, 1%, 0.5%.
- look at the E12 series:

…,
1k, 1.2k, 1.5k, 1.8k, 2.2k, 2.7k, 3.3k, 3.9k, 4.7k, 5.6k, 6.8k, 8.2k,
10k, 12k, 15k, 18k, 22k, 27k, 33k, 39k, 47k, 56k, 68k, 82k,
100k, 120k, 150k, 180k, 220k, 270k, 330k, 390k, 470k, 560k, 680k, 820k,
1M, 1.2M, 1.5M, 1.8M, 2.2M, …,

- graph it: it's logarithmic!

take 1k through 1.2k to 1.5k;

1k + 10% = 1.10k } - these two values are close enough to
1.2k – 10% = 1.08k } adjacent to cover the range at 10% tol.

1.2k + 10% = 1.32k } - these two values are close enough to
1.5k – 10% = 1.35k } adjacent to cover the range at 10% tol.

- although the values are predefined standards agreed upon back in the 60s, and have a lot of history behind them, its also a degree of good sense, due to how resistors are easiest to make relatively accurately, cheaply and durable/mass produced in factories without measuring each one and winding a wire until the right resistance

is reached etc.
- they can just machine dose out carbon or whatever powder, compress and dip in ceramic etc.

- As we have noted, there are a number of limitations to what exactly you can do with diode logic, but, some can be worked around, and, with a decent understanding of how the components work and some creative re-arranging of equations, most desired results can be achieved with sufficient circuitry.
- so, lets consider some other analogue logic circuits:

Analogue Signal Processing:

Dealing with Signal Duality - Conversion Techniques

- analogue systems have some major drawbacks - the most significant of which is the lack of decision-making.
- taking some input values and doing theoretical number crunching makes it easy to forget the actual reasons for he desired outcome
- the end goals.
- there’s more than just controlling values of outputs, i.e. creating arbitrary voltages.
- although sometimes that's great and it's exactly what you want, often it's not that exciting.

- The digital circuits looked at in depth, elsewhere on this site, on the other hand, are anything but lacking excitement, hence the ability to use a digital signal control the parameters of an analogue calculation and the ability to make digital (TRUE / FALSE) decisions based upon an analogue system would be desirable.
- This is where a range of conversion techniques, I.e. we are back to ADCs and DACs but from the analogue logic end rather than the digital POV.

Digital to Analogue Conversion

- Since the purpose of Digital to Analogue Conversion (DAC), in a broad sense, is to conditionally modify the inputs to an analogue circuit - for example, "if clock is high, set maximum limit on

tempo to 60bpm”.
- The first part of that sentence is digital (clock is either high or low) and the second part is analogue (a maximum limit on the value of tempo).

Basic Digital to Analogue Conversion:
Probably the simplest methods of Digital to Analogue Conversion is the following:

Enable function based DAC
- The enable input (on a digital sequential logic function for example) responds only to the digital component of the input signal and will act as a switch to turn **everything** inside the logic circuit on or off.
- If the digital circuit is controlling an analogue circuit then an on / off switch for the analogue circuit has been created.

- One of the useful things about an analogue circuit is that you can set the analogue value of it's output to be any integer in the [mathematical] range -100 to +100.
- This means that when it’s switched on, the analogue circuit can generate any (fixed) value desired.
- When it’s switched it off it will be stuck at 0 though.
- the the off=0% behaviour can't e directly modified, but some analogue processing can be done afterwards if need be.

- now an arbitrary analogue value can be assigned to any digital signal, or to use some technical parlance, it has been given it a "weight". This directly gets around the analogue / signal duality issue, by explicitly forcing the analogue signal to have some direct relation to the digital.

A Note On the Subtleties of Enabling Microchips

- One of the coolest features of certain digital circuits is the ability to disable and enable them at will.

- When enabled the circuit behaves normally,
- When disabled, all of the devices in the circuit turn off - any components act like they don't exist,
- vibrators stop oscillating (and their sync desyncs), logic devices stop updating and, crucially, all internal wiring goes to 0 (for both analogue and digital components).
- Note that if internal wiring goes to 0, so do all outputs from the circuit.

- Whilst the internal wiring all goes to zero, internal states of components are retained.
- So a Timer, Counter, Selector, Toggle etc., when re-enabled, will return to it's previous value.

- This highlights a couple of important system design constraints when working with components inside such logic circuits:

Firstly;
- if you create a system to create a permanent switch or set reset (or any other logic tool for state retention) that stores the current state in the wire, rather than internally to a component, then it will lose it's state when the microchip is disabled.
- For example, the following images shows two implementations of permanence:

fig 2.2.3 Permanent Switches

- The left circuit shows the switches before activation, middle is after activation and on the right is after the circuit has been deactivated and reactivated.
- In the loop-backed OR, the state is retained in the wire.
- In the counter, the state is retained internal to the component.
- For the most part, this is irrelevant - the overall behaviour is the same for both devices, but if you place them in a circuit then disable and re-enable it, then the loopback OR will reset, the counter will not (as shown by the right hand part of that image).

Secondly;
- if you have any edge-triggered devices in your system, then the re-enabling of the circuit may cause them to be re-triggered, as the signal goes from FALSE to TRUE.

Sub-note on Edge-Triggered Logic;

- In digital electronics, a device is described as edge-triggered if it updates at the instant a change of state is seen on the input, rather than just updating constantly in response to changes in the input.
- In most cases, it is the change from a FALSE value to a TRUE value, known as a rising edge - Alternatively, a transition from TRUE to FALSE is known as falling edge.

- Edge-triggered inputs include counters, toggles, select switches, certain timer inputs, all reset signals and a whole host of other thing].

- Overall, these are just design constraints.
- They could cause problems if you aren't aware of them, or they can be leveraged to your advantage.
- Sometimes it's a good thing that everything resets when the circuit is re-enabled - you can design an entire sub-circuit reset around that very premise, typically so that upon each activation of the sub-circuit it returns to a quiescent (or otherwise

predetermined) state.
- [neat!]

Analogue To Digital Conversion

- As mentioned earlier, analogue doesn't have the ability to make decisions, so the ability to generate digital signals from an analogue source is very important to get a whole lot of use out of the analogue processing tools available.

Thresholding

- The simplest form of analogue to digital conversion is thresholding;
- i.e. taking an analogue signal and saying, "if the value is greater than x, perform action".
- Now one of the reasons for explaining the comparator function of OpAmps becomes apparent!

- its also worth briefly mentioning here the existence of a special type of comparator setup called a window comparator – this is just two comparators in opposing +ve and -ve configuration so that there is an upper and lower threshold so forming a 'window' between which the comparator will trigger as high.

Comparators - Greater / Less Than

- comparators are great if you have some fixed maximum and minimum values for your range, but sometimes you will have ranges that move.
- technically, to swap between one range and another you could switch comparators quite easily, but this doesn't help if you want the activation range to be more dynamic.
- What's really needed is a method for comparing 2 analogue signals, which is again pretty simple to do. To create a system where input 1 must be greater than input 2 to activate:

$x_1 > x_2$
$x_1 - x_2 > 0$

- So, simply finding the difference between x_1 and x_2 (using subtraction), if the difference is greater than 0, then $x_1 > x_2$

- However, note, a comparator is one of those logic devices that takes the **absolute** value of its input.
- This means that all negative values will be treated as positive,

which leads to an issue with the above equation.
- In this case it means that *any* non-zero difference will trigger the output.
- So as is often the case when subtracting, we split the signal and take the positive component only, to create an analogue ‘comparator’ circuit as shown:

fig 3.2.1 Analogue ‘Comparator’ circuit

- Note the placement of the dc offset source here:
- It actually takes up the entire range of the comparator, but at exactly zero the comparator does not trigger the offset.
- As the signal has been split, this means that any negative value will also read 0, so the "greater than" comparison has been accurately created.

Comparators - Equality
- In the ‘comparator’ circuit above, an inequality test was in fact made, before adding the splitter.
- So, inverting the "not equals to" test to a "not not equal to" test, Or, if you prefer an "equal to" test we can create an equality test - simple!

- Well, sort of…
- Using that method, the difference between the two signals would have to be exactly zero, they really would have to be identical.
- However, in analogue systems, depending on what your input sources are, having two signals at exactly the same value is quite rare, So there will often be a need for margins of error on concepts like equality.

- Again, this is very simple, remove the splitter and place the offset as shown:

fig 3.3.1. Equality Comparator with 5% Margins of Error

- ‘Hanging’ the offset over the back of the comparator allows you to detect at 0% (unlike before), and up to a certain percentage.
- The above detects anything from 0-5% so as long as the absolute value of the difference between the two inputs is smaller than 5, there is "equality".
- To clarify with examples:

x_1	x_2	Output

	15	17	TRUE
	17	15	TRUE
	95	94	TRUE
	95	89	FALSE
	89	94	FALSE

- The significant point being that it doesn't matter which which is greater, as long as they are within 5% of each other then the equality condition for this circuit is satisfied.

Comparators - Dynamic Ranges

- having covered static ranges, what about dynamic ranges?
- a mixture of the techniques above will need to be used.
- In the simplest form, a dynamic range has, say, a fixed upper bound and a lower bound that can move;
- you can determine whether you are in the range by testing for each boundary condition separately and ANDing the result.
- So for upper and lower boundaries of b_{hi} & b_{lo}:

$b_{hi} > x > b_{lo}$

Which becomes:

$(b_{hi} > x) \text{ AND } (x > b_{lo})$

or, if you prefer:

$(x < b_{hi}) \text{ AND } (x > b_{lo})$

- Which is a horrible mix of analogue and digital systems all mungled into a single none too pretty equation, but should be vaguely straightforward to understand:
- To be in range you must be smaller than the high boundary and greater than the low boundary [we have created a window comparator, as mentioned earlier albeit a partly dynamic capable one hence it has a degree more complexity than the most basic form].

- However, there may be dynamic ranges that are better described by a center point, where the input has to be within a fixed distance from that center.
- This is actually the same configuration as the equality comparator circuit, but with fixed margin of error;
- i.e. a 'non-dynamic' window comparator as described initially in this section.

- Or, you might desire a range with lower boundary that moves and a fixed range, so:

$b_{hi} = b_{lo} + 15$

Where 15 is the size of the range. For this we use the following:

fig 3.4.1 Sliding Range with Fixed Width

- Noting that both upper and lower boundary are tested for inside a single ‘comparator’ circuit.
- In addition to not having the second ‘comparator’ circuit, it’s also not necessary to generate the value of the upper boundary - it's implied in this system, so no need for an adder circuit or anything fiddly like that.

- We can also have circuits where a center point and range size is input, or lower bound and size of the range is input as a variable, involving a bit of analogue processing to generate upper and lower bounds.
- Really it depends on what exactly you are testing for.
- Depending on exact needs and the nature of the dynamic range in question, more of the circuit can be pushed into the analogue part, more into the digital part or more into the ‘comparator(s)’.
- It's actually a remarkably flexible system and there are a lot of different techniques that can be employed here.

Summary

- Being able to switch between the analogue and digital signal processing models at will, and the introduction of thresholding & comparators / decision making opens up a whole world of hybrid analogue / digital logic.
- without these constraints the inputs of analogue signals aren’t limited to being simple voltage values, and the outputs no longer have to just be a simple voltage value either.

A Few Notes On Scaling

- I don’t know if this will prove useful when you can simply use an amplifier or attenuator module and adjust to needs but I’m gonna dip a little into the math just for the sake of completeness and to provide inspiration - who knows where it might be applied usefully…

Basic Scaling

- Just to clarify, when talking about "scaling" here, what is meant is multiplying or dividing an analogue signal by a *fixed* value (known as the scaling coefficient).
- So, doubling or halving the value of a signal are examples of scaling.
- The key thing to note is that it is an operation that has a single input, so is very different to more familiar multiplication and division operations, where you you would want to multiply one analogue signal by another.
- in fact it is the same thing occurring, just one signal in the equation comes from a fixed, internal [to the circuit] voltage source [the scaling coefficient].

Basic Integer Up-Scaling

- Scaling a signal up is actually quite a simple thing to do, as long as you only want to scale it by an integer (e.g. doubling, tripling, etc., not multiplying by 2.5, for example).

For example, a coefficient of three:

$$z = kx$$
$$z = 3x$$
$$z = x + x + x$$

Where k is the scaling coefficient.

- We simply add the signal to itself k times.
- fine until k becomes unreasonably large, at which point it may be significantly more efficient to find the prime factors* of the number and re-express the addition using that.
- For the example of k = 100:

$$z = kx$$
$$z = 100x$$
$$z = (2 \cdot 2 \cdot 5 \cdot 5)\, x$$

- Rather than an adder with 100 stages, instead 2 two-stage adders and 2 5-stage adders can be used, which is significantly more efficient with respect to both resources and tedious ‘wiring’.

**every natural number greater than 1 is either a prime itself or can be ‘factorized’ as a product of primes.*
The numbers greater than 1 that are not prime are called composite numbers.
Writing a number as a product of prime numbers is called a prime factorization of the number.
The terms in the product are called prime factors. The same prime factor may occur more than once.
Using this knowledge we can break scaling coefficients down into manageable parts that can be manipulated with smaller adder

blocks.

Basic Downscaling

- Downscaling is significantly more tricky that upscaling so to begin with here’s a simpler method that is essentially a cheat;
- it's pretty efficient for many signals, amongst other things working for any device where the analogue output is proportional to it's current state.

Downscaling Clocked Outputs

- considering the process for downscaling a periodic signal suhc as a clock, operating on the concept that the output is proportional to time:
- For the case of halving, we can achieve it by doubling the maximum speed of the clock and then subtracting 50% from that clock output, as shown below:

fig 1.2.2. Graphical Explanation of the solution [switch sensor to clock, radius is speed]

- from here on, scaling up along the x-axis by increasing target speed, things now get a bit complex as it really depends on what you want the clock speed to do;

- If you want it to speed upwards to 50% and then stop after T seconds then you can simply set the target time to 2T and then limit the signal to a maximum of 50%.

- The clock takes a variable input and the comparator is acting as an analogue ‘comparator’ circuit to trigger a change of that variable (using the ~~toggle~~ insert vc switch/mute) when the output is **outside** the range [0% < z ≤50%].

- Of course, as shown by the dotted line, the nature of subtraction means that we could well end up with negative value output, so we use a splitter to discard any value below zero, as shown below:

fig 1.2.3. Sensor Scaling Circuit [again, sensor to clock, radius to speed]

- As another example, If you want to create a triangle wave shaped variation in clock speed, where clock speed T rises to 50% and

then starts to descend back down, you'll have to trigger that manually [? rly], using something like the following:

fig 1.2.3. Triangle Wave Generator with 50% scaling on the output

Generalisation for any Scaling Factor

- To summarise, downscaling a signal from a [clock] by k:

1. Alter your Max [speed?] to be k times bigger than you require
2. Shift the value down so that the value at [target time] = 0 is 100/k.
3. To achieve this, for example:

        k=2, subtract 50
        k=3, subtract 66.67
        k=4, subtract 75
        …,
        k=n, subtract 100/n

Split the resultant signal and take the positive component only.

- The above will work for division by any number > 1 and is **not** just limited to integers.
- The proof is shown below:

The original line is:

$z = 100 - 100 (x/R) = 100(1 - x/R)$

The desired line is:

$z = 100(1 - x/R)/k$

Where k is the scaling coefficient and R is the maximum speed.

- So, scaling the x axis by k on the original equation:

$z = 100(1 - x/Rk)$

- then subtracting the required value to achieve the transposition;

- This is a little more complex than you might think,
- to subtract a value that is equivalent to division by k at z = 100:

$100 - y = 100/k$
$-y = 100/k - 106$
$y = 100 - 100/k$
$y = 100(1-1/k)$

So:

$z = 100(1 - x/Rk) - y$
$z = 100(1 - x/Rk) - 100(1-1/k)$
$z = 100(1 - x/Rk - 1 + 1/k)$
$z = 100(1/k - x/k)$
$z = 100(1-x) / k$

- So, since this works for any value of k, arithmetically, upscaling is also possible using this method.

Some More New Gates

- I'm now going to briefly introduce you to another couple of new logic gates
– this time they don't have any relevance in particular to a new proof i'm going to introduce or anything
– although all these gates are used in places like computing, telecommunications, cryptography and a myriad of other places, I just like weird sequencing tools and sound manglers
– so with out further ado lets look at even/odd parity gates and majority gates, along with a slightly less cumbersome way of describing their behaviour than truth tables:

- You're by now very familiar with AND and OR gates:

- **AND** gates can be considered in this type of representation:

$$AND(x_1,x_2,\ldots,x_n) = \begin{cases} 1 & \text{if all arguments are 1} \\ 0 & \text{otherwise} \end{cases}$$

- Likewise, **OR** gates can be represented as:

$$OR(x_1,x_2,\ldots,x_n) = \begin{cases} 1 & \text{if any argument is 1} \\ 0 & \text{otherwise} \end{cases}$$

- So, I present to you the **MAJORITY** gate:

$$MAJ(x_1,x_2,\ldots,x_n) = \begin{cases} 1 & \text{if strictly more arguments are 1 than 0} \\ 0 & \text{otherwise} \end{cases}$$

- And **ODD PARITY** gate:

$$ODD(x_1,x_2,\ldots,x_n) = \begin{cases} 1 & \text{if an odd number of arguments are 1} \\ 0 & \text{otherwise} \end{cases}$$

What is Parity Bit?

- The parity generating technique is one of the most widely used error detection techniques for the data transmission.
- In digital systems, when binary data is transmitted and processed, data may be subjected to noise so that such noise can alter 0s (of data bits) to 1s and 1s to 0s.

- Hence, a Parity Bit is added to the word containing data in order to make number of 1s either even or odd.
- The message containing the data bits along with parity bit is transmitted from transmitter to the receiver.

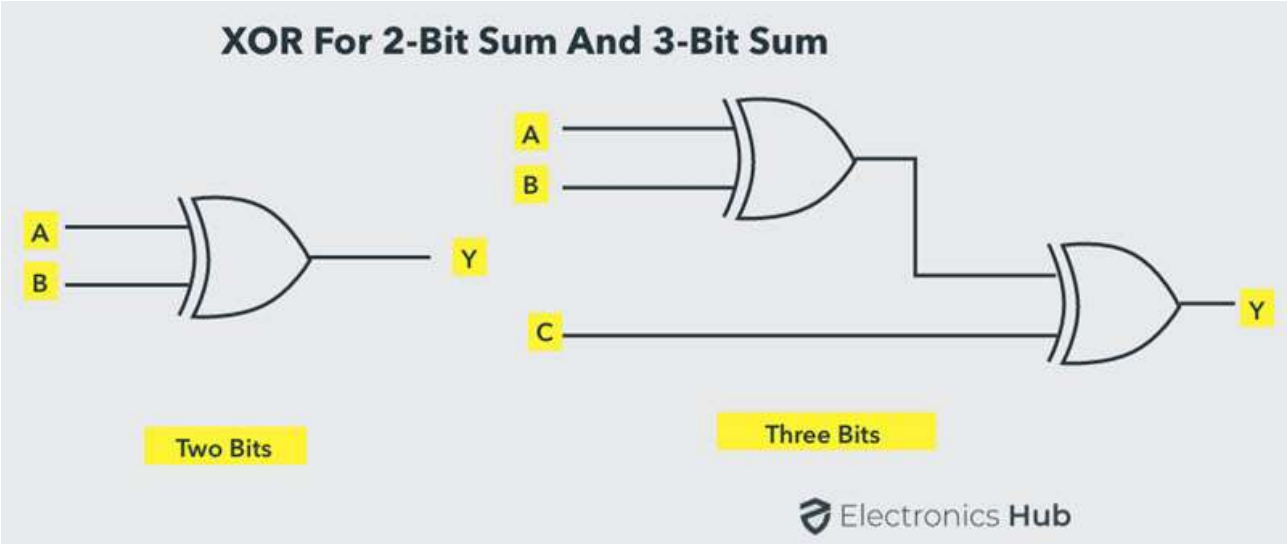
- At the receiving end, the number of 1s in the message is counted and if it doesn't match with the transmitted one, it means there is an error in the data.
- Thus, the Parity Bit it is used to detect errors, during the transmission of binary data.

Parity Generator and Checker

- A Parity Generator is a combinational logic circuit that generates the parity bit in the transmitter.
- On the other hand, a circuit that checks the parity in the receiver is called Parity Checker.
- A combined circuit or device of parity generators and parity checkers are commonly used in digital systems to detect the single bit errors in the transmitted data.

Even Parity and Odd Parity

- The sum of the data bits and parity bits can be even or odd.
- In even parity, the added parity bit will make the total number of 1s an even number,
- whereas in odd parity, the added parity bit will make the total number of 1s an odd number.
- The basic principle involved in the implementation of parity circuits is that sum of odd number of 1s is always 1 and sum of even number of 1s is always 0.
- Such error detecting and correction can be implemented by using Ex-OR gates (since Ex-OR gate produce zero output when there are even number of inputs).
- To produce two bits sum, one Ex-OR gate is sufficient whereas for adding three bits, two Ex-OR gates are required as shown in below figure.



Parity Generator

- It is combinational circuit that accepts an n-1 bit data and generates the additional bit that is to be transmitted with the bit stream.
- This additional or extra bit is called as a Parity Bit.
- In even parity bit scheme, the parity bit is ‘0’ if there are even number of 1s in the data stream and the parity bit is ‘1’ if there are odd number of 1s in the data stream.
- In odd parity bit scheme, the parity bit is ‘1’ if there are even number of 1s in the data stream and the parity bit is ‘0’ if there are odd number of 1s in the data stream. This will discuss even parity generators only to avoid duplication of effort.

Even Parity Generator

- Let us assume that a 3-bit message is to be transmitted with an even parity bit.
- Let the three inputs A, B and C are applied to the circuit and output bit is the parity bit P.
- The total number of 1s must be even, to generate the even parity bit P.

- The figure below shows the truth table of even parity generator in which 1 is placed as parity bit in order to make all 1s as even when the number of 1s in the truth table is odd.

3-bit message			Even parity bit generator (P)
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0

1	0	0	1	
1	0	1	0	
1	1	0	0	
1	1	1	1	

From the above truth table, the simplified expression of the parity bit can be written as

$$P = \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,B\,C$$

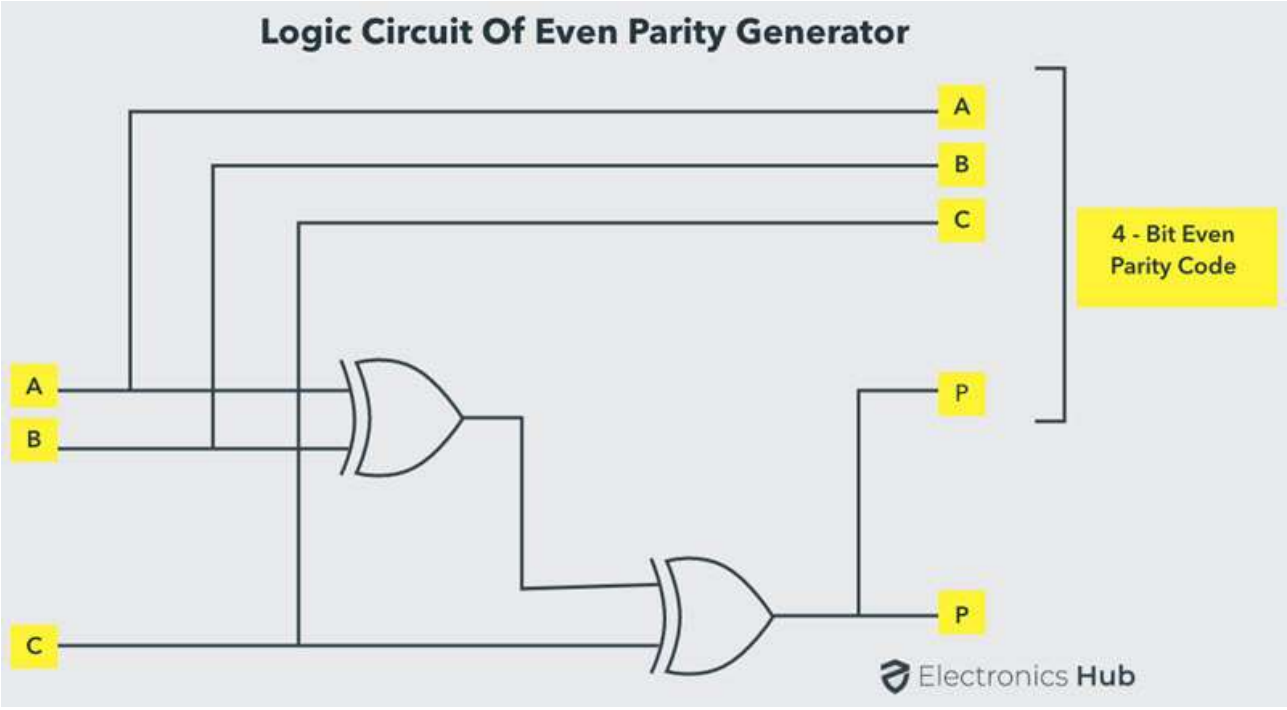
$$= \bar{A}\,(\bar{B}\,C + B\,\bar{C}) + A\,(\bar{B}\,\bar{C} + B\,C)$$

$$= \bar{A}\,(B \oplus C) + A\,(\overline{B \oplus C})$$

$$P = A \oplus B \oplus C$$

- The above expression can be implemented by using just two Ex-OR gates.
- The logic diagram of even parity generator with two Ex – OR gates is shown below.
- The three bit message along with the parity generated by this circuit which is transmitted to the receiving end where parity checker circuit checks whether any error is present or not.

To generate the even parity bit for a 4-bit data, two Ex-OR gates are required to add the 4-bits and their sum will be the parity bit.



Even Parity Checker

- Consider that three input message along with even parity bit is generated at the transmitting end.
- These 4 bits are applied as input to the parity checker circuit, which checks the possibility of error on the data. Since the data is transmitted with even parity,
- four bits received at circuit must have an even number of 1s.
- If any error occurs, the received message consists of odd number of 1s.
- The output of the parity checker is denoted by PEC (Parity Error Check).

- The below table shows the truth table for the Even Parity Checker in which PEC = 1 if the error occurs,
- i.e., the four bits received have odd number of 1s and PEC = 0 if no error occurs,
- i.e., if the 4-bit message has even number of 1s.

4-bit received message				Parity error check
A	B	C	P	Cp
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

- then deriving the boolean expression from the table:

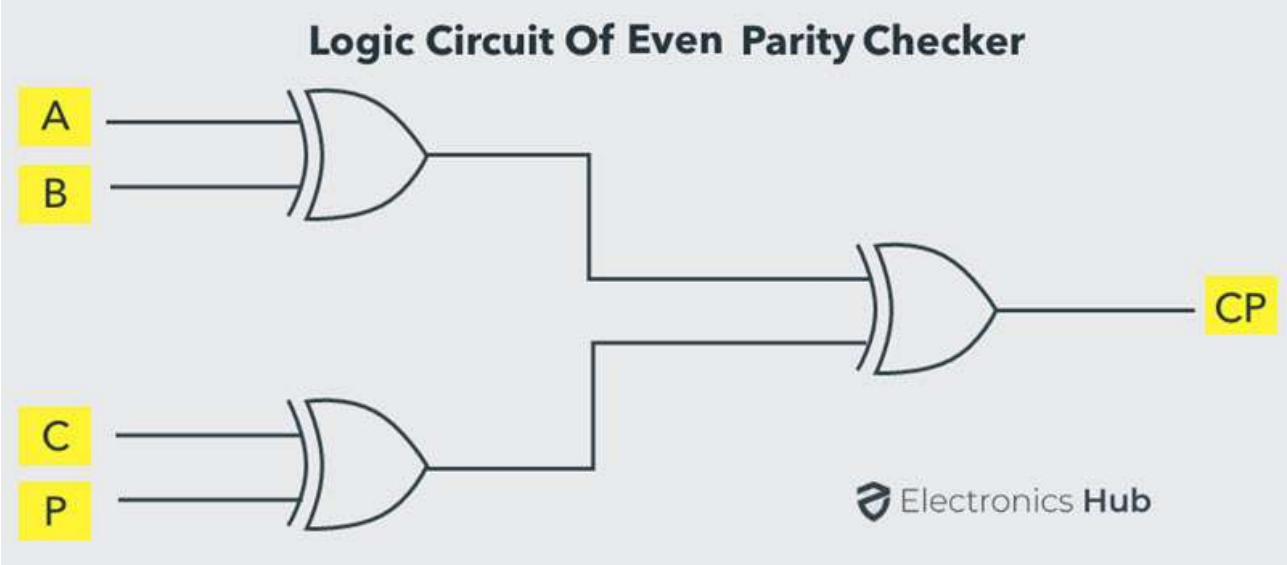
$$PEC = \overline{A}\ \overline{B}\ (\overline{C}\ P + C\overline{P}) + \overline{A}\ B(\overline{C}\ \overline{P} + CP) + AB(\overline{C}\ P + C\overline{P}) + A\overline{B}\ (\overline{C}\ \overline{P} + CP)$$

$$= \overline{A}\ \overline{B}\ (C \oplus P) + \overline{A}\ B(\overline{C \oplus P}) + AB(C \oplus P) + A\overline{B}\ (\overline{C \oplus P})$$

$$= (\overline{A}\ \overline{B} + AB)(C \oplus P) + (\overline{A}\ B + A\overline{B})(\overline{C \oplus P})$$

$$= (A \oplus B) \oplus (C \oplus P)$$

- The above logic expression for the even parity checker can be implemented by using just three Ex-OR gates as shown in figure.



- This whole set of truth tables, boolean expressions and logic circuits also can be generated for odd parity generators and checkers similarly of course, but i'm going to omit those because it's not really going to demonstrate anything not already shown above.

- by now you're probably starting to realise it's pretty much possible to design a gate with near any possible truth table imaginable;
- and often they can be built in analogue logic form as well as digital.

- more obscure gate combinations with far weirder outcome requirements can be required – let's see the approach to designing the gate combinations required to achieve them:

- **How do we achieve** a gate combination that produces the outcome '**At least k**'?

- as we know already:
- A n-way majority circuit takes n inputs and returns 1 if, and only if, at least strictly more of its inputs are 1 than 0.
- **Given an n-way majority circuit**, we can **describe how to build an At-Least-k circuit** that **returns 1 if and only if at least k of its inputs are 1**. how?

- **considering a 2n-way majority circuit,** if:

n-k+1 of the **inputs are set to = 1**,
k-1 of the **inputs are set to = 0**, **and,**
the **original n inputs are our signal inputs.**

Giving $z = MAJ(x_1, \ldots, x_n, c_1, \ldots, c_n\text{-}k+1, b_1, \ldots, bk\text{-}1)$

$z = MAJ(x1, x_2, x_3, c_1, b_1, b_2)$

- that's all very well as a leap from question to answer with no process, but how do we actually get from the question to that answer?
- and, how do we get to the actual circuit diagram for this?
- lets take, for example n = 3, and k = 3:

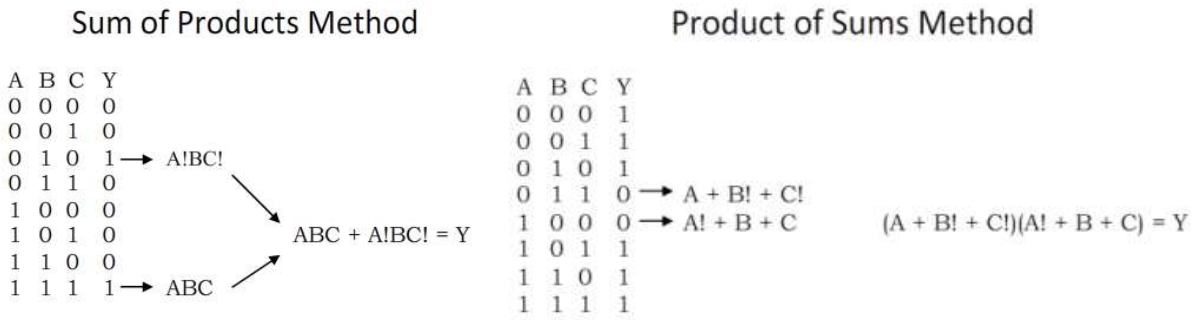
1. First of all draw a truth table:
- put **all possible** combinations of **input values** based on chosen number of inputs

x_1	x_2	x_3	c_1	b_1	b_2	$z \geq k$ [≥ 3]	req'd	check
0	0	0	1	0	0	0	0	y
0	0	1	1	0	0	0	0	y
0	1	0	1	0	0	0	0	y
1	0	0	1	0	0	0	0	y
0	1	1	1	0	0	1	constant ≤ 1	y
1	1	0	1	0	0	1	constant ≤ 1	y
1	0	1	1	0	0	1	constant ≤ 1	y
1	1	1	1	0	0	1	1	y

- next enter the **outcomes**
- *do the calculated [z ≥ k, i.e. ≥ 3] match the required outcomes?*
- every 0 and 1 in the [z ≥ k] and req'd columns match so the

checks are sound.

2. Convert truth table to boolean expression:



- we have an equal number of 0s and 1s in our output column so the choice of method is moot but generally one would choose the least:

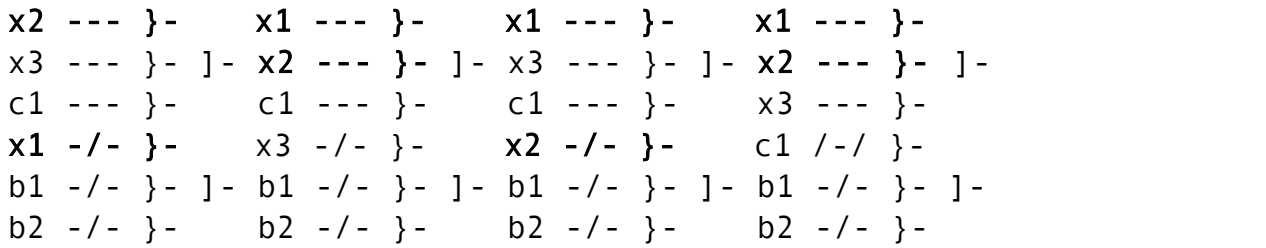
$$x_1!x_2x_3c_1b_1!b_2! + x_1x_2x_3!c_1b_1!b_2! + x_1x_2!x_3c_1b_1!b_2! + x_1x_2x_3c_1b_1!b_2! = z$$

3. Finally convert boolean expression to logic circuit:

- The technique for converting boolean expressions to logic circuit symbols follows this order:

1. **Bracketed quantities**
2. **NOTs**
3. **ANDs**
4. **ORs**

- there are no **bracketed** quantities
- then grouping the **NOTs** and those that aren't inverted;
- then **ANDing** together those groups of NOTs and uninverteds
- finally, **ORing** together the groups of ANDs and ORs:

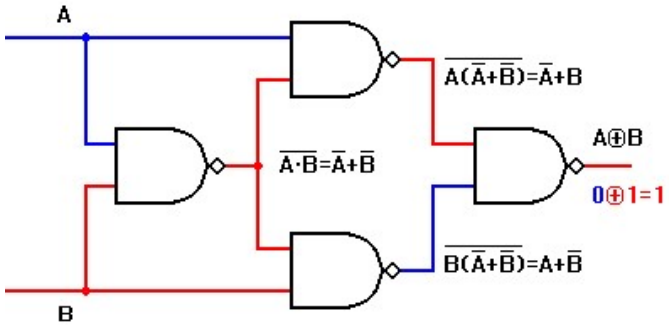


- **resulting diagram can then be refined** by breaking down into nands or swapping redundant gates for more efficient ones.

for example;

- since **all gates in boolean algebra can be constructed from**

NANDs; for example, an **XOR** from NAND gates:



The boolean expression for output is as below

$$Y = \overline{\overline{(A \cdot \overline{(AB)})} \cdot \overline{(B \cdot \overline{(AB)})}}$$

Let's simplify it using DeMorgan's theorem

$$\begin{aligned}
&= \overline{\overline{(A \cdot \overline{(AB)})}} + \overline{\overline{(B \cdot \overline{(AB)})}} \\
&= (A \cdot \overline{(AB)}) + (B \cdot \overline{(AB)}) \\
&= A \cdot (\overline{A} + \overline{B})) + B \cdot (\overline{A} + \overline{B}) \\
&= A\overline{A} + A\overline{B} + B\overline{A} + B\overline{B} \\
&= A\overline{B} + B\overline{A} \qquad \cdots (\because X\overline{X} = 0) \\
&= \mathbf{A} \oplus \mathbf{B}
\end{aligned}$$

Any gate can be built from NAND or NOR gates
- As well as making an XOR gate, NAND gates can be combined to create any type of gate.
- This enables a circuit to be built from just one type of gate.
- For example an AND gate is a NAND gate then a NOT gate (to undo the inverting function).

- To change the type of gate, such as changing OR to AND, you must do three things:

1. Invert (NOT) each input.
2. Change the gate type (OR to AND, or AND to OR)
3. Invert (NOT) the output.

- For example an OR gate can be built from NOTed inputs fed into a NAND (AND + NOT) gate.

- the reason for doing this is to reduce the number of ic's used in a circuit;
– when only one or two of multiple types of gates are used then 3 or 4 different quad ICs might be reduced to just 2 or 3 quad NANDs,
- so although more total individual gates now form the circuit the total cost in ICs is reduced
– saving space, wiring,, increasing efficiency, reducing failure risk and saving money.

---

so,

- recreating the diagram long form in NANDs:

NAND NOT x_2 x_3 c_1 | NAND NOT x_1 x_2 c_1 | NAND NOT x_1 x_3 c_1 | NAND NOT x_1 x_2 x_3 | OR [2xNOT>NAND]] 4 input ---- --] OR 2 input = z

NAND x_1 b_1 b_2 | NAND x_3 b_1 b_2 | NAND x_2 b_1 b_2 | NAND c_1 b_1 b_2 | OR [2xNOT>NAND]] 4 input ---- --] OR input

= 8 x 3 input NANDs + 17 x 2 input NANDs = 3 x 4023 + 5 x 4011 = 8 ICs

- but any NOTs following eachother – i.e. directly adjacent, can be removed as a pair:

3 in AND AND AND AND > 4 input OR] > 2 input OR = z
3 in NAND NAND NAND NAND > 4 input OR]

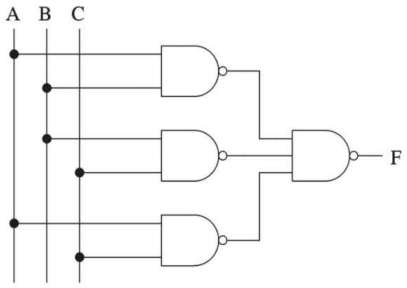
= 2 x 4023 + 2 x 4073 + 1 x 4072 + 1 x 4001 = 6 ICs

- you will likely be left with a simplified condensed circuit requiring a different number of ICs than the NAND based circuit that does the desired task.
- note, the circuit may or may not be more efficient as built purely from NANDs as condensed to specific gate types.
- In this case specific gate types uses 2 ICs less than all NANDs and so is the superior choice.
- *[although when it comes to mass production using entirely NANDs*

may still be advantageous due to the savings in bulk buying only NANDs vs. 3 or 4 different IC types].

- As a result, we have constructed a 6 input majority gate in the most efficient manner possible from 2 x4073s, 2 x 4023s, a 4072 and a 4001 to give the out put z.
- we have designed a 6 input MAJ from basic AND, NAND and Ors!

- comparing our final logic circuit to the 3 input MAJORITY circuit shown to the right:



- the similarities in structure are quite apparent which is a big clue that the initial suggestion that the answer is a 6 input MAJORITY circuit with 3 constants is in fact correct.

- Likewise to design an '**Exactly k**' circuit, we need a circuit that **outputs 1 if and only if exactly k of its inputs are 1**.

- consider:
- **If k = n**, **an n-way AND gate is used**.
- Otherwise, **building up an 'At-least-k'** circuit as before, and **a modification of it, an At-least-k+1** circuit, using the same techniques as in the previous example.
- It **has exactly k inputs that are 1** if the first circuit **outputs 1**, but **the second one outputs 0**.

```
z = AND(At-least-k, At-least-k+1)
  = AND(MAJ(x1, x2, x3, c1, b1, b2), (MAJ(x1, x2, x3, c1, b1, b2)+1))
```

- The full truth table and circuit diagram for this would be somewhat enormous so I wont draw out the full calculation as a proof.
– if you really need to see it, have a go!

- With this kind of approach you can solve near any sequencing, drum pattern, event triggering, melody switching or whatever patch-programming issue you face purely by analysing it logically and using some basic utilitarian tools.

Analogue Sort

- There's endless uses for sorting values as with any other logic and therefore likewise with analogue logic the equivalent is even more true as ever
- not only is sorting values useful but the resulting cv shaping and audio wave form outputs that can be derived etc. are great.

- the most basic sort function is to order the inputs by value based on breaking the problem down to smaller and smaller comparisons - otherwise known as 'divide and conquer' sorting.
- As has been shown, max() and min() enable finding the highest and lowest values from a set of numbers this is also the first step in sorting.

Finding the median Value:

- The method of finding the middle of three value involves the following process:

1. Find all pairs of signals
2. Get the min() value for each pair.
3. Take the max() of these min()s.

- The concept here is that if you take all possible pairs, you are guaranteed to have one pair that consists of the highest value and the second highest value.
- This particular pair will return a min() value equal to the second largest value of the three and none of the other min()s can return a larger value.

- First, identifying the pairs of signals, and their minmum values to give the three intermediary values:

$y_1 = \min (x_1, x_2)$

$y_2 = \min (x_1, x_3)$

$y_3 = \min (x_2, x_3)$

Then finding the maximums of those values:

$z = \max (y_1, y_2, y_3)$

- Two relevant points of note, although I've stated this is the middle value of three, it is in fact the method for finding the second largest of a set of values
- i.e. expanding the number of inputs does not continue to give the median with this method but the second largest.

- also, note that when input values are duplicates, the sort does not actually find the second largest value,
- it actually sorts the values in descending magnitude and the output is the value of the second index of that ordered list.
- i.e. generally the output is the actual second largest so the sorting will be correct, but for a sort, "second largest value" is slightly misleading.

Finding the Second Smallest Value:

- A very similar way can be used to find the second smallest value by replacing the max() function for a min() and the min()s for max()s.
- Although this is redundant with three inputs middle value being middle value, this is relevant when using more inputs:

- This 'merge sort' method works for any number of inputs, the number of pairs just increases - instead of only 3 pairs to deal with for 3 inputs, with four inputs 6 pairs are needed:

$(x_1, x_2), (x_1, x_3), (x_1, x_4), (x_2, x_3), (x_2, x_4), (x_3, x_4)$

- The formula for the number of pairs, which allows us to calculate the number of gates needed, is as follows:

$p = [n (n-1)] / 2$

where
p is the number of pairs required
n is the number of inputs

- Since this is a 2nd order polynomial (related to the square of n) the sort method becomes exponentially more complex as the number of inputs increases.

Sorting into ordered lists:

- Since min, max, second smallest and second largest value can be found, a circuit can be made to take 4 analogue input values and output the same four values in descending order (or therefore

technically any order you patch them).

Finding the nth Largest / Smallest Value:
- it's also possible to select the value that is nth largest / smallest value in a set;
- as with finding the 2nd largest value, values were grouped into pairs, for 3rd largest they are grouped into triplets and for the nth largest they are grouped into groups of size n.

- Although the complexity increases even more steeply with input and desired output out complexity it allows the generation of a fully ordered list of values or selecting a specific value from that list.
- This solves the earlier problem of finding the true Median value from a set.

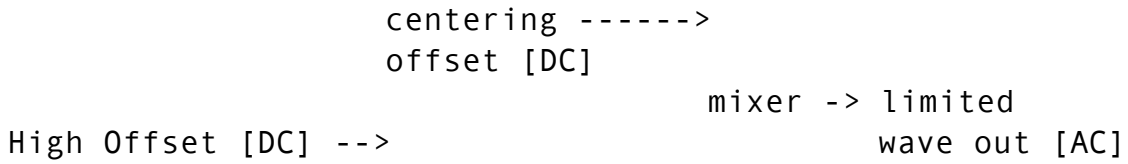
- There are also methods of sorting based on comparison sort algorithms, which can also be achieved with min and max functions.

- Basic forms of comparison sort exist, such as Bubble sort, sometimes referred to as 'sinking sort', that repeatedly steps through a list comparing adjacent elements and swaps them if they are in the wrong order, passes through the list is repeating until the list is sorted.

- these comparison sorts are not suitable for finding the value of a specific index in the list, without actually ordering the entire thing, such as the Median, but may be simpler than the number of gates required for high input numbers.

Example application of analogue logic circuitry:

- 'Limiting'; using simple waveform manipulation circuitry to convert a triangle wave to a sine wave using devices discussed in this document:



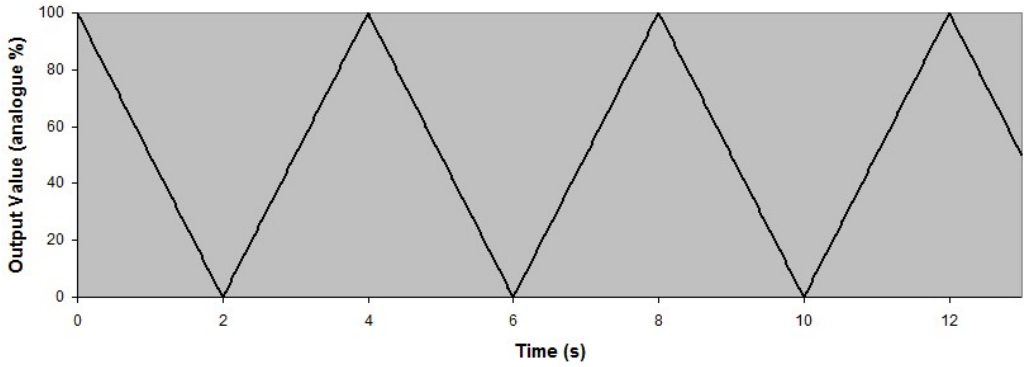
```
wave                        AND ------------>
in [AC]     -->
                OR ------>
Low         -->
offset [DC]
```

Signal sources

- In this case, there's an AC input tri wave and some DC offsets as the signal sources.
- The tri wave has the following properties:

    Period: 4*s (note, this is arbitrary)*
    Shape: Triangle

as per the graph below:



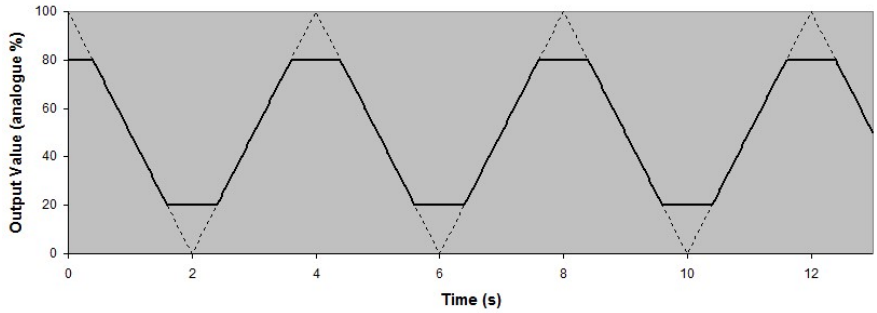
4.2. Signal Limiting

- As discussed above, some of the simplest operations we can do on analogue signals is min() and max(), using ANDs and ORs.
- We can use this to achieve some boundary limiting on the signal, preventing the signal going above 80% or below 20%.

- So taking the input tri wave and using an OR gate with a DC offset giving a constant 20% will give us a max function, which leads to a *lower* boundary.
- To clarify: if the wave in is greater than 20, the output of the OR gate is equal to the wave, but if the wave drops below 20, then the OR gate will output 20.

- After this we do the same thing with an AND gate to do a min() with an 80% DC offset, to give an upper limit, and the resulting output of the AND gate is shown below (with the original wave in dotted lines).

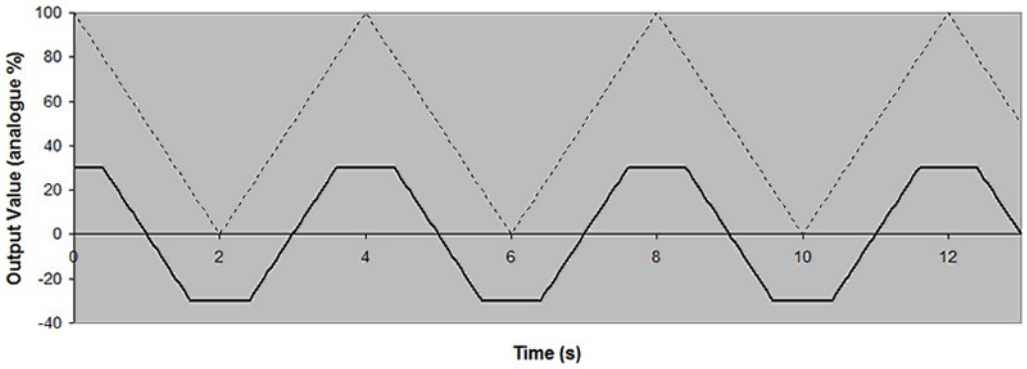
- You can see that the timer's output is ignored when it exceeds the minimum and maximum bounds that we have set for it:



- Conceptually it is a little unintuitive that we use a min() function to create a maximum value, and vice versa, but that's how it works I assure you.

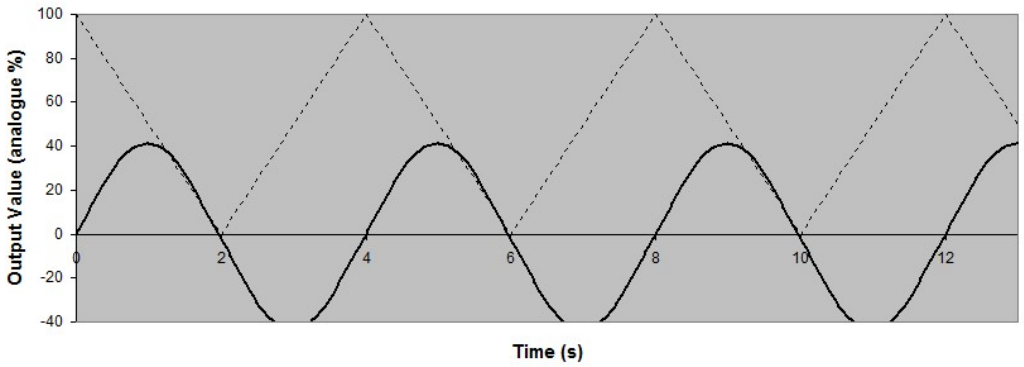
Removing the DC Offset

- Sometimes, especially with waveforms, it's useful to have them centred around 0, and to do this we need to shift the whole wave down by 50 (the point at which it is currently centred - its average value).
- This is a simple subtraction operation using a summing mixer and a SC offset providing 50%, results are as shown:



Extending the concept

We can further process this waveform by integrating it - actually turning out to be a rough approximation of a sine wave, as shown below:



Note: *This is only an approximation of a sine wave.*

Ive thrown in the math below for those interested but its not particularly significant in any special way.

constant c_3 ---->

c_2 -------> sum ---> trapezoid

max() ----------->

Tri ---->

min() ->

c_1 ---->

If c_1 is 20, c_2 is 80 and c_3 is 50

At any given time, the voltage of the limited waveform output an be calculated by the equation:

$V = Max(Min(Vtri, c_1), c_2) + c_3$

finally integrating the trapezoid waveform will give a very good approximation of a sin wave.

$f(x) = (2/\pi) \sin\text{-}1[\sin(\pi x)]$

Glossary And Reference

here is a definitive list of logic operations and their strict definitions – they're presented in this form so that unambiguous outcomes for any truth table can be calculated:

- **NOT**(x_1) = {1 if argument is 0
  {0 otherwise

- **AND**$(x_1,x_2,\ldots,x_n)$ = {1 if all arguments are 1
  {0 otherwise

- **OR**$(x_1,x_2,\ldots,x_n)$ = {1 if any argument is 1
  {0 otherwise

- **XOR**$(x_1,x_2,\ldots,x_n)$ = {1 if either one, but not both nor none of arguments are 1
  {0 otherwise

- **MAJ**$(x_1,x_2,\ldots,x_n)$ = {1 if strictly more arguments are 1 than 0
  {0 otherwise

- **ODD**$(x_1,x_2,\ldots,x_n)$ = {1 if an odd number of arguments are 1
  {0 otherwise

- **IMPLY**$(x_1,x_2,\ldots,x_n)$ = {1 if 1 unless first argument is 1 and second argument is 0
  {0 otherwise

- **CONV**$(x_1,x_2,\ldots,x_n)$ = {1 unless only second argument is 1
  {0 otherwise

[inverted operation equivalents like NAND, NOR and XNOR omitted]

- more...?

- *...*