Phys. 334 **Day 6 Lecture Notes**

Introduction to digital electronics and positive feedback

Analog: Information is proportional to a continuos range of voltages or currents. For example, 1.2 V means something different than 1.4 V.

Digital: Information is the presence or absence of a signal pulse. There are only two possible states of any signal, typically called 1 and 0. Usually, these states are represented by voltages. For example, 5 V might represent the 1 state and 0 V the 0 state.

Although digital (binary) systems are much less efficient for storing information than analog systems, there is important advantage of digital systems over analog systems: noise immunity. If the two digital states are so different that they cannot be confused for one another, one can store information (and computer programs) that will not change in time: the information can be safely stored and manipulated. In analog systems, the presence of electrical noise means that the information is forever changing: what is 1.2 V now may look like 1.4 V later when noise is present.

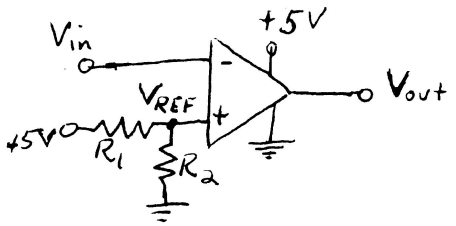
A key ingredient of any digital system is a definition of two distinct states and the means to always distinguish between the states. For example, if our two states are 0 V (plus some noise) and 5 V (plus some noise), how do we determine whether an input signal, V_{in}, is in one state or the other. This is where an op-amp without negative feedback comes in handy.

Recall that the high gain of an op-amp means that if $V_+ > V_-$, V_{out} will saturate to $\approx V_{CC}$, while for $V_+ < V_-$, V_{out} will saturate to $\approx -V_{EE}$.

For $R_1 = R_2$, $V_{REF} = 2.5$ V, so for

$V_{in} < 2.5$ V, $V_{out} \approx 5$ V and for

$V_{in} > 2.5$ V, $V_{out} \approx 0$ V.



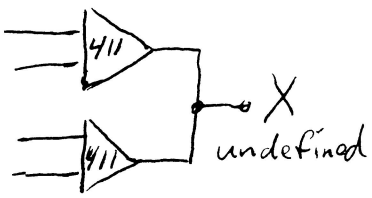
Note, there is no negative feedback here.

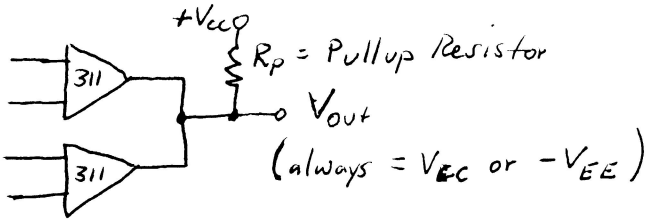
This circuit is called a **comparator**: as long as the noise on V_{in} does not exceed ≈ 2 V, a 5 V input signal will not be confused for a 0 V signal. Any old op-amp can serve as a comparator, but the slew rate limits how fast the output can change as the input changes. Because we want to transfer digital information at high rates, special op-amps, called comparators, like the 311 you will use in the lab, have been developed that have 3 nice features:

1.) High slew rate (and very short time to come out of saturation)
2.) Single ended power (ie they can operate off a single positive power supply, with $-V_{EE}$ set to ground, and the output swings all of the way to $-V_{EE}$.)
3.) Open collector output (you provide a 'pull-up' resistor)

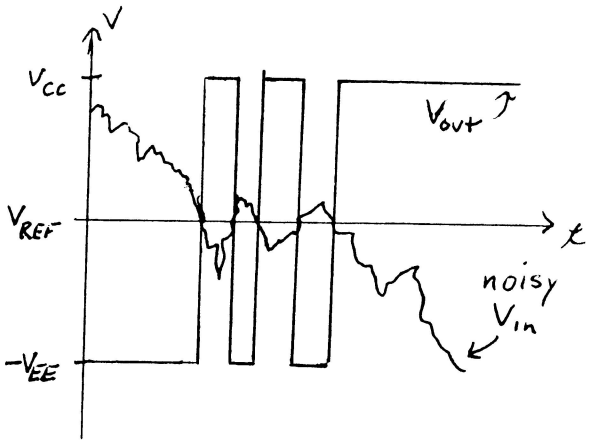
The third feature, open collector output, requires some explanation. It means that when the output is low (at $-V_{EE}$), the output has a very low output impedance (like a normal op-amp, so that current can easily flow into the output). But when the output is high, the output has a very high impedance (like an open circuit, so no current flows). The reason this is desirable is because very often, you need to connect the outputs of several digital circuits together. For

example, in computer systems, the outputs of a variety of devices must send data to the central processing unit (CPU) on shared data lines. You cannot connect the outputs of two normal op-amps together: they would fight and the resulting voltage would be unpredictable. On the other hand, you can connect the outputs of two open collector op-amps together: if either or both of the outputs is low, current flows through the pull-up resistor making $V_{out} = 0$ V, and only if both outputs are high will no current flow, making V_{out} high.





Comparator op-amps are so fast that they present a problem: as V_{in} changes from low to high or high to low, there will always be some noise on the signal as shown to the right. When V_{in} crosses V_{REF}, noise on V_{in} will cause several crossings and each will cause the output to change. That is, a single change in state of V_{in} can cause a number of fast (high frequency) pulses on V_{out}. Fast pulses are deadly for digital systems because parasitic capacitive coupling allows the rogue pulses to appear everywhere.



As always, there is an elegant solution to this noise generated problem: add some positive feedback to the comparator circuit. This makes a circuit called a **Schmitt Trigger**.

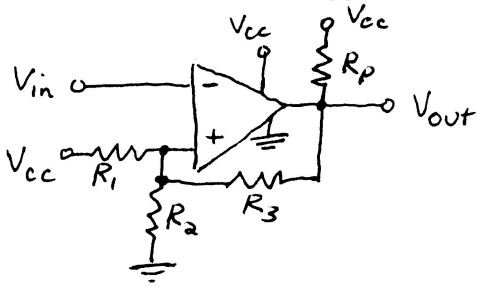
$V_+ = V_{REF}$ now has two values depending on the state of V_{out}.

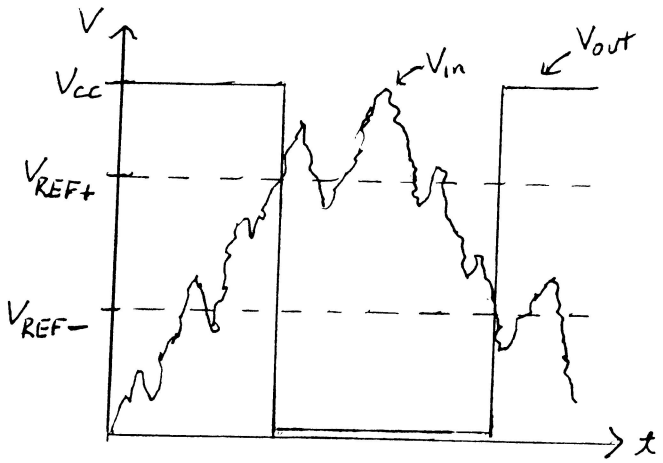
When V_{out} is high (open circuit)
$V_+ = V_{REF+} = V_{CC} \times R_2/((R_P + R_3)||R_1 + R_2)$

When V_{out} is low (0 Volts)
$V_+ = V_{REF-} = V_{CC} \times R_2||R_3/(R_1 + R_2||R_3)$

and $V_{REF+} > V_{REF-}$.

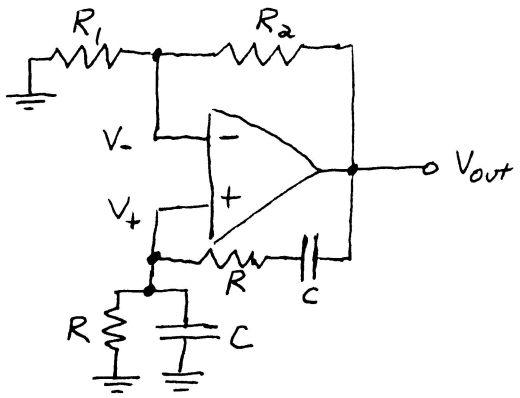
Now, when V_{in} starts low, V_{out} is high and $V_+ = V_{REF+}$. The first time V_{in} crosses above V_{REF+}, V_{out} switches low and V_+ jumps down to V_{REF-}, well below the noise level on V_{in}. The reverse happens when V_{in} passes from high to low. You choose your R's so that $V_{REF+} - V_{REF-} >$ noise level on V_{in}.





For many of the digital chips that we will use, the inputs have Schmitt trigger circuits built into them to prevent noise from feeding through. Chips with Schmitt trigger inputs are represented by the symbol:  at the inputs (representing the hysteresis created by the positive feedback).

Positive feedback can also be used to create an oscillator. The **Wien Bridge Oscillator** is a nice example and is one of the few ways you can create a sine-wave from scratch.



Here we have an op-amp with no input and both positive and negative feedback. Let's assume that the output will not saturate (so $V_+ \approx V_-$) and see if that is possible.

$V_-/V_{out} = R_1/(R_1 + R_2)$ and

$V_+/V_{out} = Z_1/(Z_1 + Z_2)$ where $Z_2 = R + 1/j\omega C$ and

$Z_1 = R||Z_C = R/j\omega C/(R + 1/j\omega C) = R/(1 + j\omega RC)$

Now for $V_+ = V_-$, we need $1/V_+ = 1/V_-$ or $1 + R_2/R_1 = 1 + Z_2/Z_1$, or $R_2/R_1 = Z_2/Z_1$. But,

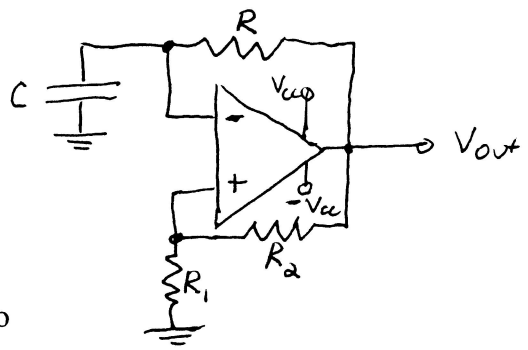
$Z_2/Z_1 = [(1 + j\omega RC)/j\omega C]/[R/(1 + j\omega RC)] = (1 + j\omega RC)^2/j\omega RC = 2 + (1 - \omega^2R^2C^2)/j\omega RC$

Which means we need: $R_2/R_1 = 2 + (1 - \omega^2R^2C^2)/j\omega RC.$

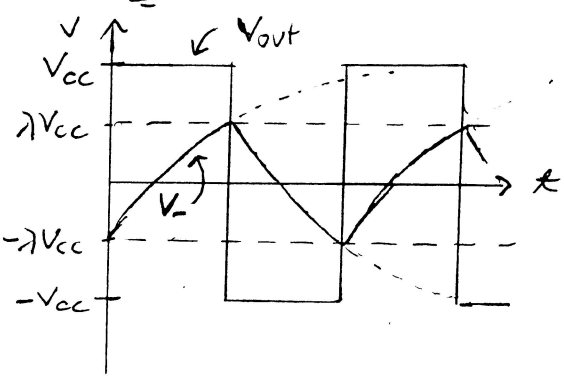
This is only possible if $R_2 = 2R_1$ and $1 - \omega^2R^2C^2 = 0$, or $\omega = 1/RC$.

So we choose $R_2 = 2R_1$ and we find that the circuit will oscillate at an angular frequency, ω, which means the output will be a sine-wave at that angular frequency. To guarantee that R_2 is exactly equal to $2R_1$, we make one of the resistors variable (eg use a lamp where the resistance depends on the amount of current -- the current will vary until $R_2 = 2R_1$).

If you don't need a sine-wave output, you can make a **relaxation oscillator** that has a square wave output. Again, the circuit has no input and both positive and negative feedback.



When $V_{out} = V_{CC}$, $V_+/V_{out} = R_1/(R_1 + R_2) = \lambda$ and the capacitor, C, will charge up through R until $V_- > V_+ = \lambda V_{CC}$. At this instant, V_{out} will switch to $-V_{CC}$ and V_+ will switch to $-\lambda V_{CC}$ and C will discharge through R until it reaches $-\lambda V_{CC}$ which will cause V_{out} to switch to $+V_{CC}$ and the cycle repeats.

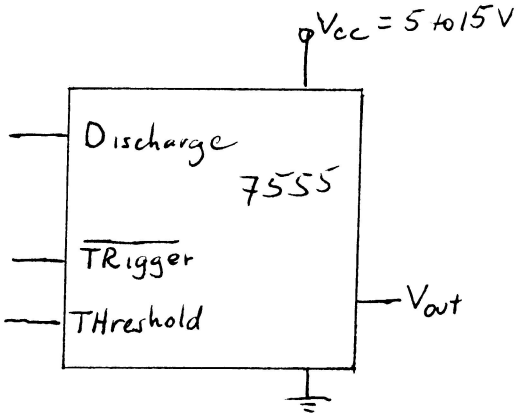


If we approximate the section of the exponential charging of the capacitor as a straight line (valid

for small λ), we have $dV/dt = I/C$, and $dV = 2\lambda V_{CC}$ and $I \approx V_{CC}/R$, we get $dt = CdV/I \approx 2\lambda RC$. This is the time for one-half of the cycle, so the period becomes $2dt \approx 4\lambda RC$ and the output frequency is one over the period or $f \approx 1/(4\lambda RC)$.

The complete calculation that includes the exponential charging of the capacitor gives a similar result: period $= 2dt = 2RC \ln[(1 + \lambda)/(1 - \lambda)]$ which reduces to our approximation when λ is small.

A general theme in digital electronics is that if there is a function that is widely needed, there will a chip available that does it for you. If this case, rather than build our own relaxation oscillator to give us a square wave at a desired frequency, we would just use an oscillator chip such as the **555 Timer Chip**. In the lab, we will use a newer version of the 555, the 7555, which has the same function but allows the output voltage to swing all of the way from V_{CC} to ground.



1.) V_{out} is bistable (digital). It is either V_{CC} or 0.

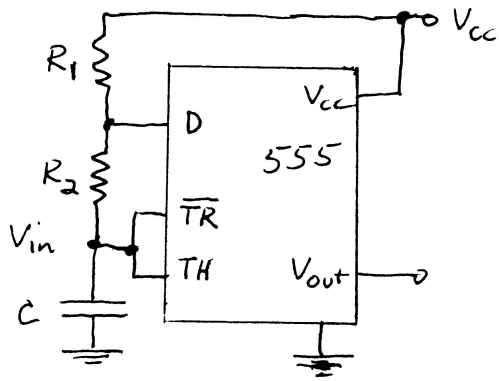
2.) Trigger ($\overline{TR}$) and Threshold (TH) are inputs (usually connected together):

$\overline{TR}$ compares its input to $V_{CC}/3$ and if $V_{in} < V_{CC}/3$, V_{out} goes high (to V_{CC}) and D (Discharge) becomes an open circuit.

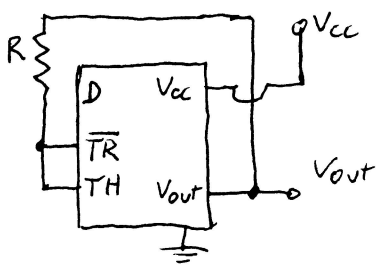
TH compares its input to $2V_{CC}/3$ and if $V_{in} > 2V_{CC}/3$ V_{out} goes low (to 0) and D becomes short-circuited to ground.

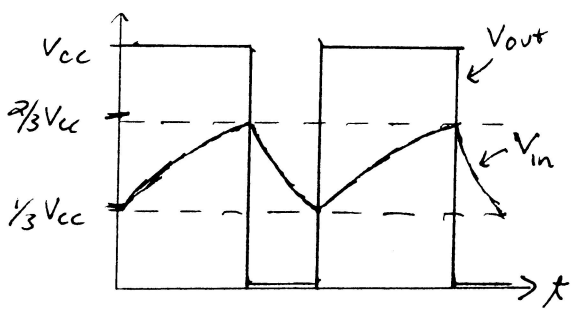
When $V_{CC}/3 < V_{in} < 2V_{CC}/3$, V_{out} and D do not change -- they stay in whatever state they were in.

One way to use the 555 is shown to the right: if V_{in} starts low, then V_{out} is V_{cc} and D is open-circuited so C charges up through $R_1 + R_2$, with time constant $(R_1 + R_2)C$. When V_{in} reaches $2V_{CC}/3$, V_{out} switches to 0, D gets connected to ground, and C discharges through R_2 to ground (through D). When V_{in} reaches $V_{CC}/3$, V_{out} switches to V_{CC} and the cycle repeats. The period for one cycle becomes $(R_1 + 2R_2)C \ln(2)$.



To make a symmetric square wave output, you can forget about D and use V_{out} to charge and discharge C through the same R, as shown below.





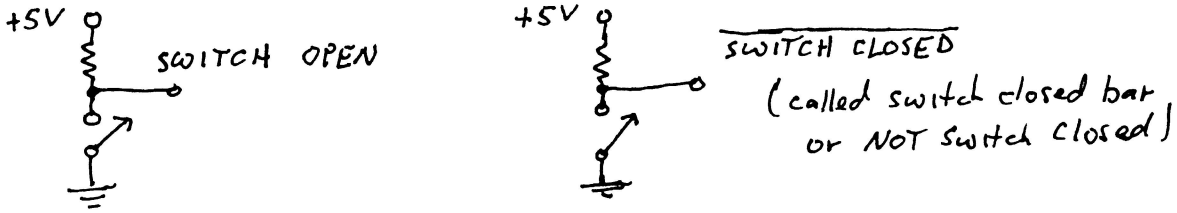
Digital Electronics

Digital signals are defined to have one of two values, called 0 or 1. A single digital value is called a 'bit'. The digital state is often a voltage, most commonly 5 V and 0 V, but in different devices, it can be any property that has distinct values. For example, on hard drives or floppy disks, the digital states are values of magnetization, for optical disks they are states of reflectivity.

The labeling of the states as 0 or 1 corresponds to their behavior under Boolean Algebra (0 = False, 1 = True). Either of the 2 physical states can represent the Boolean algebra state: when the higher voltage (eg. 5 V) represents 1, this is called 'positive true' logic while if the lower voltage state (eg. 0 V) represents 1, it is called 'negative true'. These conventions are summarized in the table below.

State	Boolean Algebra	Voltage (positive true)	Voltage (negative true)
first	1 = True	High (eg. 5 V)	Low (eg. 0 V)
second	0 = False	Low (eg. 0 V)	High (eg. 5 V)
third	----------	Open-circuit	Open-circuit

We can minimize confusion by labeling digital signals descriptively. The convention is that if the signal is negative true, then a bar is placed over it, as illustrated below.



The 'third state' referred to in the table above is not a logical state. It refers to the a property of many digital chips that allow the output to be disabled, ie not asserted. This allows the outputs of chips to be connected together: as long as only one chip is on (the others in the open-circuit state), the output is well defined. Chips that have this open-circuit feature are referred to as 'tri-state'. (Tri-state is like the open-collector output we saw last week, except the output is open-circuit for either value of the input.)

Numbers and bases:

A string of bits such as 1011 can be interpreted as a number in base 2, such like we interpret $327 = 3 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$ in base 10:

$11001 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 25_{10}$

You can even have decimal fractions like 511.078 (by continuing with 2^{-1}, 2^{-2} ...) but this isn't very common.

To convert from decimal to binary, just successively divide by 2:

25/2 = 12, remainder 1 and you read off the remainders in reverse order:
12/2 = 6, remainder 0
6/2 = 3, remainder 0
3/2 = 1, remainder 1
1/2 = 0, remainder 1 $= 11001_2$

To avoid having to write down long sequences of 0's and 1's, it is customary to use hexadecimal (base 16) notation: give each group of 4 bits a single hexadecimal symbol:

0000 = 0	0001 = 1	0010 = 2	0011 = 3
0100 = 4	0101 = 5	0110 = 6	0111 = 7
1000 = 8	1001 = 9	1010 = A	1011 = B
1100 = C	1101 = D	1110 = E	1111 = F

(0-9 are just the binary equivalent of the base 10 digits, A-F are extra)

so for example, 1001000101101100 would become 1001 0001 0110 1100 = $916C_H$.

eg. $9E_H = 9 \times 16^1 + (E=15_{10}) \times 16^0 = 144 + 15 = 159_{10}$

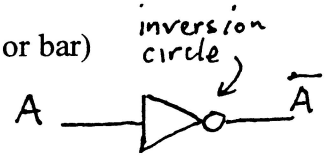
Hex notation is convenient because it saves space and most computer systems store information in 'bytes', groups of 8 bits, which can then be represented as 2 Hex digits. ASCII (American Code for Information Interchange), which is a standard for sending information between systems, uses 7 bits for a character with the 8th bit used as a 'parity' bit for error checking.

There are other conventions for digital numbers described in section 8.03 in your text. You should read about these although we won't talk about them in class.

Logic Gates:

A logic gate is a device whose output(s) is a logical function of its present inputs. They comprise the building blocks of all digital electronics. All logical functions are combinations of the following one and two input operations. They are defined via positive true convention: 1 = High = H, 0 = Low = L:

INVERT (Not, or bar)



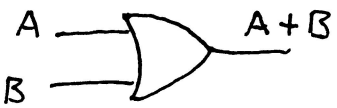
Truth Table

A	$\bar{A}$
H	L
L	H

Note

$\overline{(\bar{A})} = \bar{\bar{A}} = A$

OR (+)



A	B	A+B
L	L	L
L	H	H
H	L	H
H	H	H

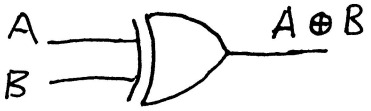
$A+B = B+A$

AND (•)

A	B	A•B
L	L	L
L	H	L
H	L	L
H	H	H

$A \cdot B = B \cdot A$

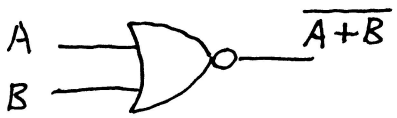
XOR, (EXCLUSIVE OR) ($\oplus$)



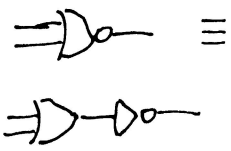
A	B	A⊕B
L	L	L
L	H	H
H	L	H
H	H	L

$A \oplus B = B \oplus A$

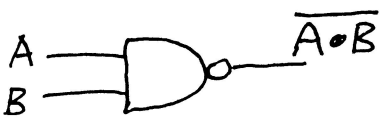
NOR ($\overline{+}$)



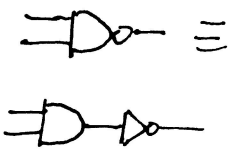
A	B	$\overline{A+B}$
L	L	H
L	H	L
H	L	L
H	H	L



NAND ($\overline{\bullet}$)



A	B	$\overline{A \cdot B}$
L	L	H
L	H	H
H	L	H
H	H	L



Note, the circle, ○, in the symbols always represents  = invert.

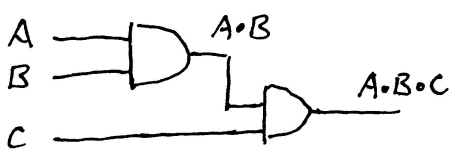
Table 8.3 (p. 491) in your text has a list of theorems for the logic operations that should, for the most part, be obvious. In general, we can replace any logic gate with a combination of other gates.

To make an inverter (NOT gate) from a NAND or NOR gate, just tie the inputs together:

To make an AND gate from NAND gates:

This is useful because gates come in groups on a chip, 6 NOT gates on a 7404 chip, 4 NAND gates on a 7400 chip. You often have spare gates left over in your circuit -- take advantage of them to avoid having to add another chip.

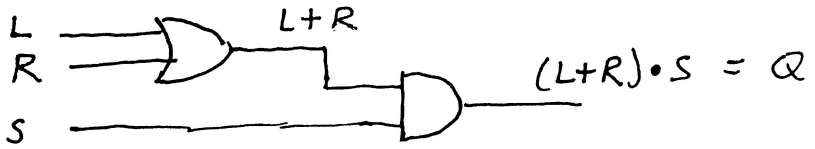
You can make 3-input gates by combining 2-input gates: eg 3-input AND gate:



A	B	C	A•B•C
L	L	L	L
L	L	H	L
L	H	L	L
L	H	H	L
H	L	L	L
H	L	H	L
H	H	L	L
H	H	H	H

Gate example 1: You want a circuit that will sound a car buzzer if either the left (L) door **or** right (R) door is open **and** the driver is seated (S): $Q = (L + R) \bullet S$

For positive true signals, the logic circuit would be the one shown below.

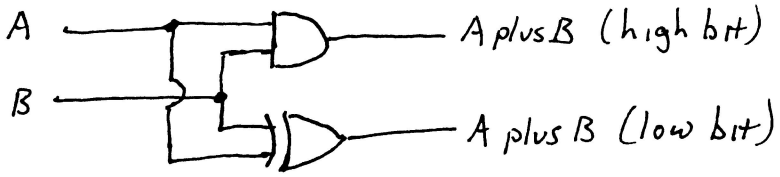


Gate example 2: Use gates to construct a device that adds two 1-bit numbers, giving a 2-bit result: $0 + 0 = 00$, $0 + 1 = 01$, $1 + 0 = 01$, $1 + 1 = 2 = 10_2$

A	B	A plus B
0	0	0 0
0	1	0 1
1	0	0 1
1	1	1 0

AND ↑ ↑ XOR

We make the truth table and recognize the left (low bit) column of Q to be an XOR gate and the right (high bit) column to be an AND gate.



Assertion Level Logic Convention

Our logical gates (and physical chips) are defined for positive true inputs (ie 1 = H, 0 =L). It is often the case that your signals are negative true (for example, if the door and seat sensors in Gate Example 1 above were 0 V when a door was open or driver seated), then what gates do you use to find Q?

The answer lies in De Morgan's theorems and assertion level convention.

De Morgan's theorems are: $\overline{A} + \overline{B} = \overline{A \bullet B}$, $\overline{A} \bullet \overline{B} = \overline{A + B}$

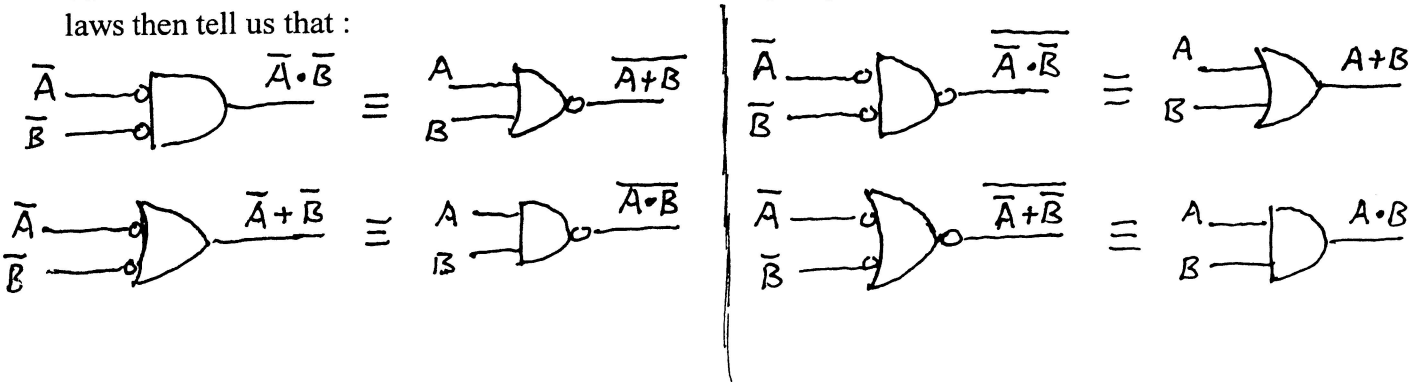
eg.

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$	$A \bullet B$	$\overline{A \bullet B}$
0	0	1	1	1	0	1
0	1	1	0	1	0	1
1	0	0	1	1	0	1
1	1	0	0	0	1	0

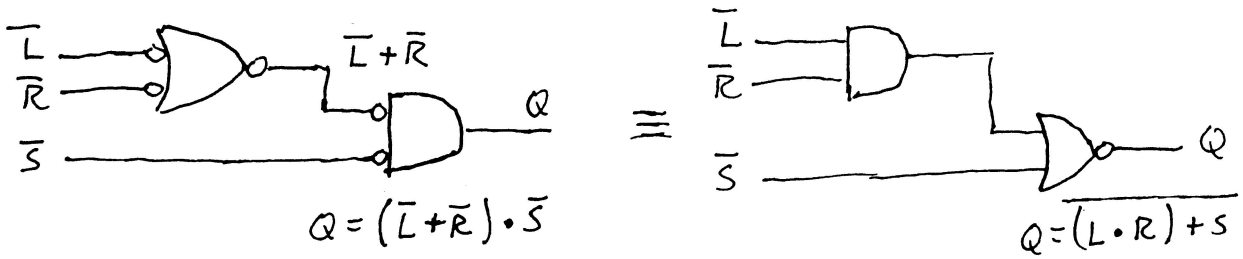
is a truth table proof of the first De Morgan Law

(ie the OR of negative true inputs is the positive true NAND function, etc).

Assertion level convention is to use the normal gate shapes for the logic function you need, but to include the inversion circle at both ends of every negative true signal. De Morgan's laws then tell us that :



Gate example 1 with negative true inputs: (L, R, S)



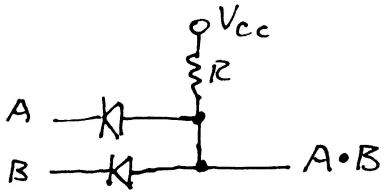
You can also prove this with Boolean algebra:

$$Q=(\bar{L}+\bar{R})\cdot\bar{S} = (\overline{L\cdot R})\cdot\bar{S} = \overline{L\cdot R+S}$$

$$\bar{L}+\bar{R}=\overline{L\cdot R} \qquad \bar{A}\cdot\bar{S}=\overline{A+S} \quad \text{where } A=\overline{L\cdot R}$$

Logic Families:

First, let's make an AND gate with resistors and diodes:



The output is high (V_{CC}) only if both inputs are high (V_{CC}), because then, both diodes are off. If either or both of the inputs is low (0 V), then that diode is on and the output will be 0.6 V. This circuit has a problem in that we cannot use its output to be the input of another diode gate: the low output voltage (0.6 V) is not the same as the low input voltage.

To make a consistent family of gates, we need the input states to be the same as the output states so we can connect outputs to subsequent inputs. There are many logic families that do this. By far, the two most common families are **TTL** (transistor-transistor logic) and **CMOS** (complementary-metal-oxide-semiconductor, a type of field effect transistor).

Each family is self consistent but **you cannot, in general, mix chips between these families**. TTL has gone through many generations. The surviving version is LS TTL, eg. 74LS00, which is what tou will use in the lab.

LS TTL: Needs +5 V ± 5% for V_{CC},
Low means ≤ 0.8 V (typically 0.4 V), High means ≥ 2.0 V (typically 3.4 V)

A low input requires ≈ 0.4 mA to flow out of the input.
A high input requires no current flow (high impedeance)
A low output can accept (sink) up to 4 mA of current
A high output has an output impedance of ≈ 120 Ω.
Power is consumed for both states.
Speed ≈ 25 MHz

Some consequences: fanout is ≈10 (one output can drive 10 inputs).

You cannot connect a TTL input to ground through a resistor larger than ≈ 1k or else the current flowing out of the input may raise the voltage above the allowed level. Inputs left unconnected (never a good idea) appear as High inputs.

CMOS: Can operate with V_{CC} between 3V and 15 V. Some families are made to be similar to TTL and need ≈ 5 V for V_{CC}.

Low means ≤ $V_{CC}/3$, High means ≥ $2V_{CC}/3$

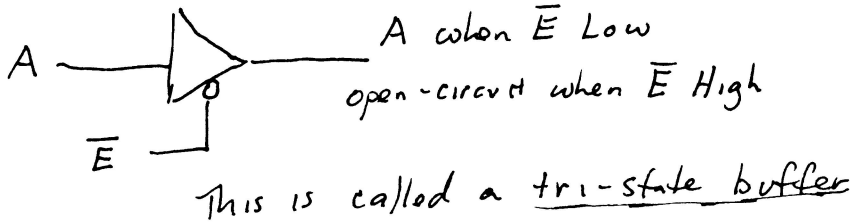
Input current (either state) = 0.
Output impedance (either state) ≈ 100 Ω
No power consumed in either state (only when switching between states).
Speed ≈ 2 - 100 MHz (depending on family)
Fanout ≈ 10 (at maximum speed -- current needed to charge input capacitance).

You can never leave unused CMOS inputs disconnected -- they will pick up stray charge and cause the outputs to switch wildly (which consumes power and maybe fry the chip).

The reason you cannot mix TTL and most CMOS chips is that a high TTL output is only guaranteed to be ≥ 2.0 V (typically 3.4 V) while a CMOS input requires ≥ $2V_{CC}/3$ ≈ 3.4 V to be interpreted as a high. CMOS outputs can, on the other hand, usually drive TTL inputs.

Tri-State Outputs:

Many CMOS gates come with an additional input, called 'ENABLE' (E or $\bar{E}$). When E ($\bar{E}$) is true, the chip acts like a normal gate. When E ($\bar{E}$) is false, the ouput of the chip is disabled, ie put in a high impedance state so that it neither tries to go high or low. This feature allows the outputs of different tri-state gates to be connected together, as long as you are careful to only enable one chip at a time.



This is called a tri-state buffer