Electronics

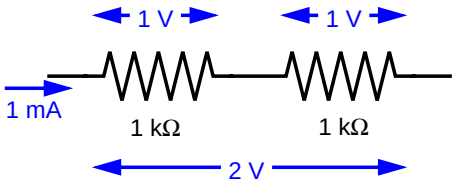
This is NOT a course in electronics.

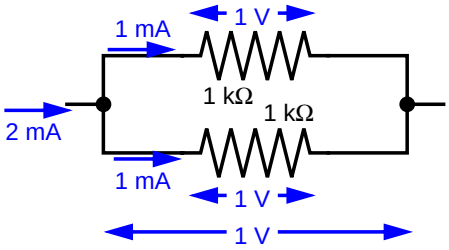
But

- Some VLSI implementation is influenced by the underlying circuit design.

This lecture is intended to:

- Outline the (few) 'need to know' facts





e.g. Ohm's and Kirchoff's Laws

Series resistors add $R = R_1 + R_2 + \ldots$

Parallel resistors 'divide' $\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} + \ldots$

- Give some 'flavour' of the underlying technology

A rough analogy

Those of you who know some physics can skip this section unless you want to read it for amusement. Those who do not may find it useful to get a 'picture' of how electricity behaves.

Compare an electrical circuit to a system of water flowing through pipes (and some other structures).

If there is a *flow* there is a **current**. A higher current means more water/electricity passes a point in a given time. The bulk measurement of the *amount* of electricity is called **charge**. Electric current is the movement of charge/time. Charge is measured in coulombs (C) and current in amps (A). One amp is one coulomb per second.

To make a current flow requires some pressure or **voltage** (measured in volts – V)[1]. A higher pressure (voltage) will push along a higher current. *N.B. volts do not 'flow' despite the popular misconception; they just 'push'.* Opposing this flow there is some friction or **resistance**; increasing the resistance will reduce the current (flow) for a particular voltage. Resistance is measured in ohms (Ω). Voltage, current and resistance are related by **Ohm's Law**:

$$V = I \times R$$

Thus, doubling the voltage across a fixed resistor will double the current. Doubling the resistor across a fixed voltage will halve the current.

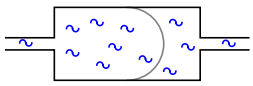
The other (simple) electrical law is **Kirchoff's Law** which outlaws leaks. Thus is so much charge (water) flows into a wire (pipe) junction the same amount must flow out.

That's about it, except there are some other effects which can be of concern. Only one is of immediate interest here, which is capacitance.

1. Unless you have a superconductor, which we don't.

Capacitance

Very loosely, capacitance is a measure of how reluctant something is to change its voltage. A component with high capacitance will require a lot of charge to flow in (or out) to make a significant voltage change.



In the water-flow analogy, a capacitor is a bit like a reservoir; the 'level' (voltage) tends to want to stay the same. It can be though of as a rubber sheet across the flow which stretches and pushes back harder as it is distorted.

Every component, including wires, have a certain amount of capacitance; thus it takes some effort to make their voltage change. In VLSI design it is typically the voltage which represents the '0's and '1's and, therefore, switching these takes effort.

Capacitors

Capacitors are devices specifically intended to store charge. They are occasionally made on-chip in analogue circuits but are almost always parasitic in digital circuits. The only real exception is that they may be used to stabilise the power supply.

Capacitance (C) is defined by: $Q = C \cdot V$

Note that the voltage is proportional to the stored charge.

Note: unlike resistance, *parallel capacitances add together.*

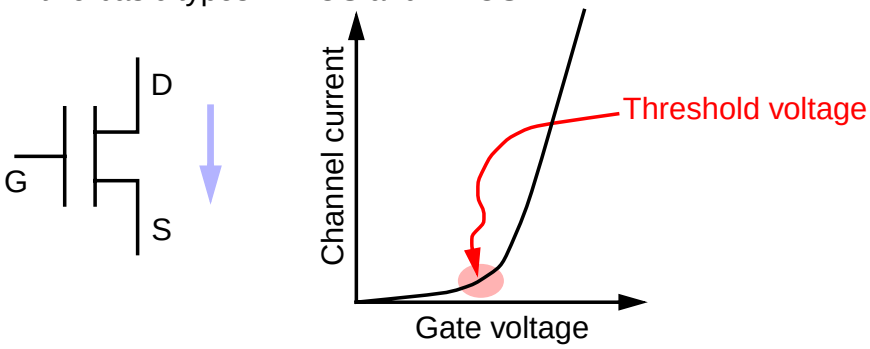
Impedance

Like resistors, capacitors 'impede' current flow although in a different way. The general term for this effect is 'impedance'. The impedance of a resistor is just its resistance. The impedance of a capacitor is more complex (depending on the frequency of the signal).

Transistors are different again; however 'impedance' will be used to indicate how they affect flow and you can think of it as 'sort of like resistance'.

Transistors

MOSFETs come in two basic types: NMOS and PMOS



Both are three terminal devices {Gate, Source, Drain}

- NMOS transistors are:
  - turned 'on' by a positive (high) gate voltage
  - good at pulling low
- PMOS transistors are:
  - turned 'on' by a negative (low) gate voltage
  - good at pulling high

Acting together in a Complementary way these transistors can provide effective two-way switching.

Transistors

The transistors used in the vast majority of VLSI implementation are Field-Effect Transistors (**FET**s). FETs come in different forms but we are concerned with metal-oxide-semiconductor field-effect transistors (MOSFETs). The term refers to the construction of the transistor which originally comprised of layers of metal and semiconductor, separated by an insulating 'oxide' layer.

This term is strictly inaccurate now as the original 'metal' layer is usually not made of metal and, more recently, the 'oxide' is often replaced with better ('high-k') materials such as hafnium silicate or zirconium silicate. However the term remains (and is likely to persist).

The terminals are called the '**gate**', '**source**' and '**drain**'. The gate is the control input. A 'channel' connects the source and drain. Technically *charge carriers* flow from source to drain (charge carriers are negative (electrons) in an NMOS FET and positive ('holes' – i.e. the lack of electrons) in a PMOS FET) but the transistor can be regarded as symmetric.

Although the FET can be used as an analogue device, in the majority of VLSI they are simply regarded as switches, being conducting ('on') or non-conducting ('off'),

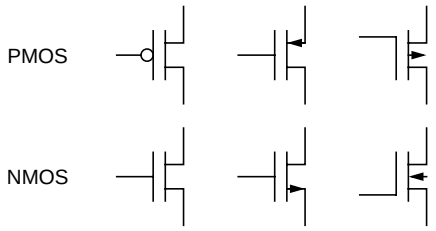
Current flows in the channel if the transistor is 'on'. An NMOS FET is 'on' if its gate is 'high' *relative to at least one of the source or drain*; similarly a PMOS FET is 'on' if its gate is 'low'.

> More specifically, as the gate voltage is raised (lowered) the channel becomes more conductive and at some point – known as the transistor's *threshold voltage* – 'switching' is regarded as having occurred.

FETs are voltage-driven: i.e. they are controlled by the voltage of the gate (relative to the other connections). This means that no current flows to or from the gate except when switching when the (capacitive) gate is charged or discharged. Therefore (to a good approximation) CMOS circuits dissipate power only when switching.

Transistor symbols

There are some different symbols which are used for MOSFETs

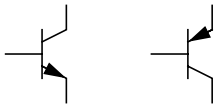


We will use the left-most symbols in each case: these show the capacitive character of the gate input and the active-low (PMOS) or active-high (NMOS) state of the control input.

(The fourth connection on the right-most symbols represents the semiconductor 'bulk' material the transistor is embedded in. It is not usually of interest to us so we can ignore it hereafter.)

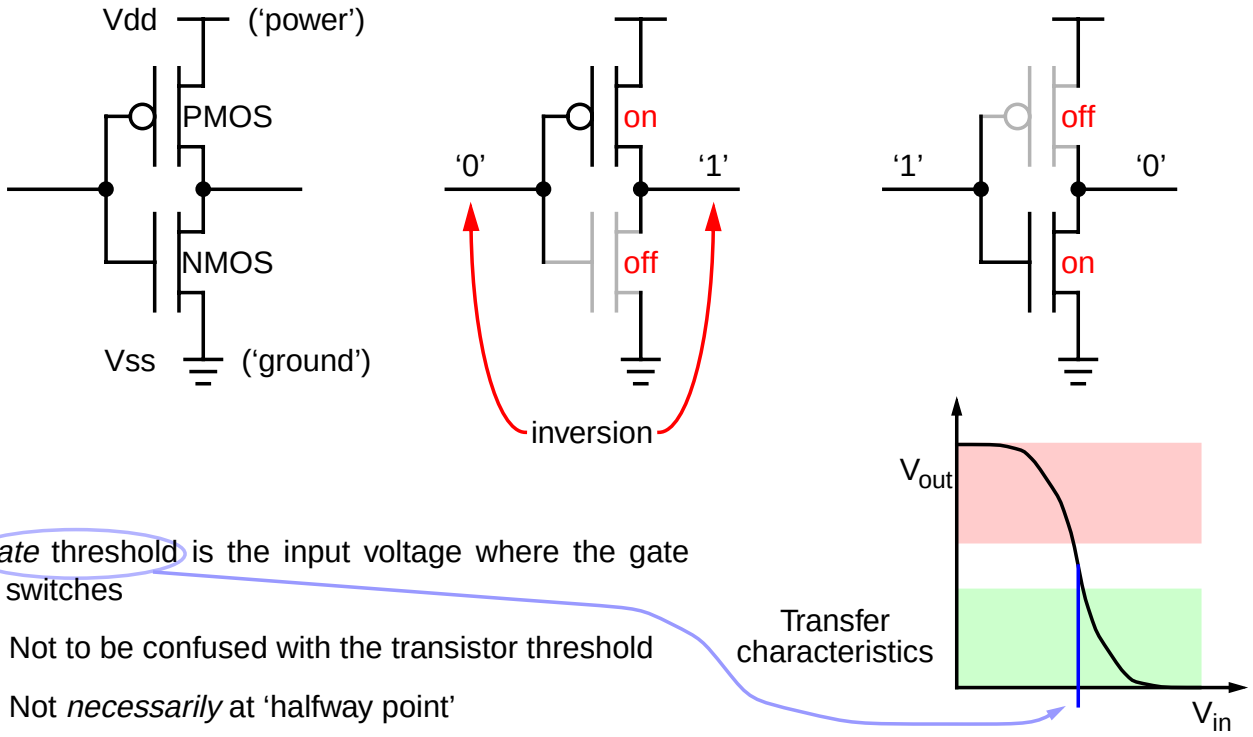
Other transistors

Another, different family of transistors are **bipolar transistors**. These are (largely) no longer used for logic circuits as they have higher power dissipation, although they have advantages in other circumstances. They are mentioned here to avoid misinterpretation and later confusion. Other than that you may want to forget about them!



The CMOS inverter

The inverter is the simplest CMOS structure:

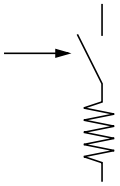


The *gate* threshold is the input voltage where the gate output switches

- Not to be confused with the transistor threshold
- Not *necessarily* at 'halfway point'

Gate *Model*

A simple model of a transistor (MOSFET) is a voltage controlled switch in series with a resistor.



This is not strictly accurate because the *channel impedance* is not an Ohmic resistor, but it is adequate as an illustrative model.

This means that the output – which is primarily capacitive – is charged and discharged through an impedance which limits the current flow … and hence the voltage change.

Consider a capacitor C charged to voltage V_0 then discharged through a resistor R by closing a switch.

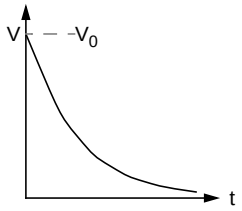
$I = \frac{V}{R}$ $I = \frac{dQ}{dt}$ $Q = C \cdot V$

so: $\frac{V}{R} = I = \frac{dQ}{dt} = -C \cdot \frac{dV}{dt}$ (the '-' sign is because the voltage diminishes as the charge flows off the capacitor)

This solves as: $V = V_0 e^{\frac{-t}{RC}}$

i.e. the (dis)charge is exponential.

The *time constant* is R × C; increasing either slows down the 'edge'.



This is *close* to the case with a CMOS gate.

Power

First, some definitions:

- Energy (E) is the capacity to do work
- Power (P) is the rate of working: i.e. $P = \frac{E}{t}$

There is little need for fancy maths to work out the energy in switching a load. Instead consider a full cycle $0 \Rightarrow 1 \Rightarrow 0$.

A node is charged with $Q = C \cdot V$ and then discharged again. There is a total transfer of Q from one power supply to the other, a potential V. The energy transfer is therefore: $E = Q \cdot V = C \cdot V^2$ regardless of how long it takes.

The energy 'cost' of cycling that node therefore depends on the capacitive load and the *square* of the supply voltage. The power therefore depends on this and also the switching frequency; faster switching dissipates more power.

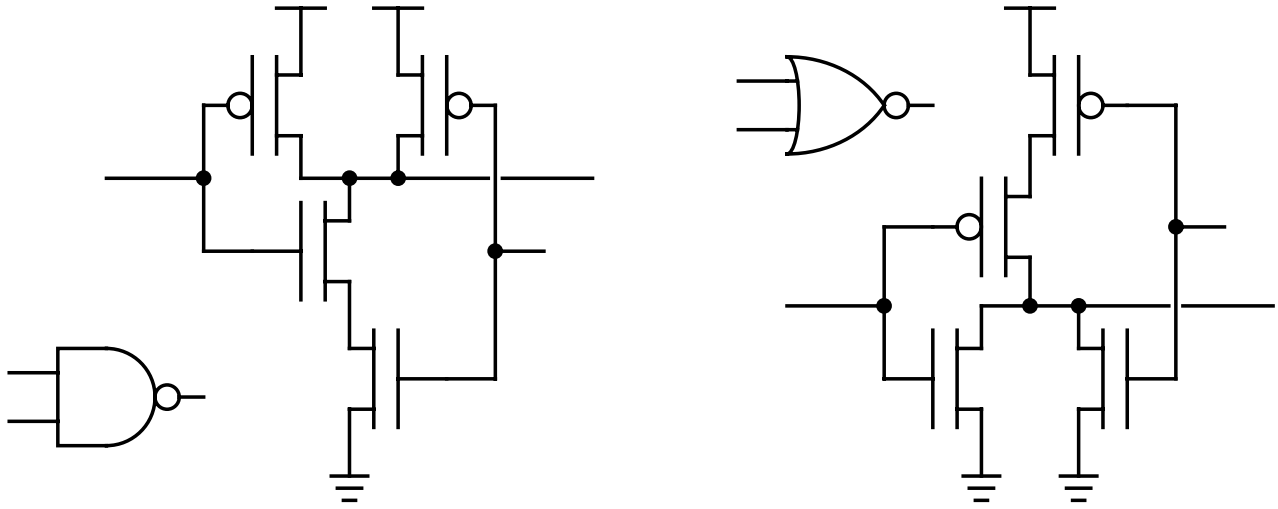
More precisely …

In a real gate the load switching tends to dominate the power but there are other causes of power dissipation:

- during switching both P- and N-transistors may (briefly) be 'on' together, so some charge is lost through this route.
- transistors do not turn 'off' completely so there is some *leakage* power at all times.
  - leakage is currently small in most cases, but getting increasingly significant as geometries shrink. A serious future worry.
  - proportionately, the biggest leakage tends to be in SRAM where there are many transistors which rarely switch.

CMOS gates

Gates are formed using serial and parallel combinations of transistors:

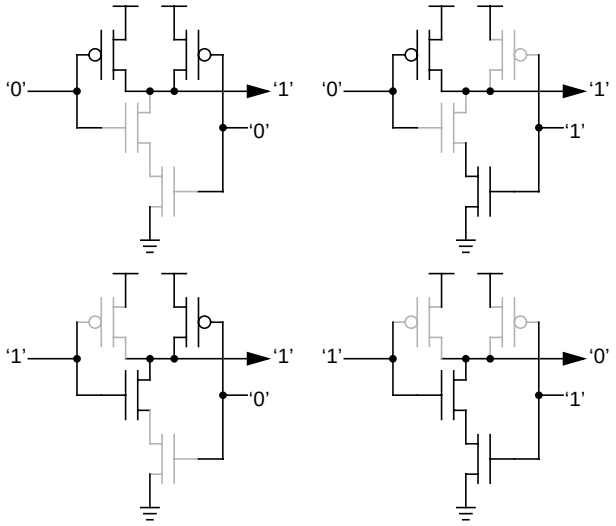


Basic CMOS gates are always **inverting** (NOT, NAND, NOR)

They provide a **load** on their inputs and actively **drive** their output.

CMOS NAND operation

The figure below shows the four possible input states of the two input NAND:

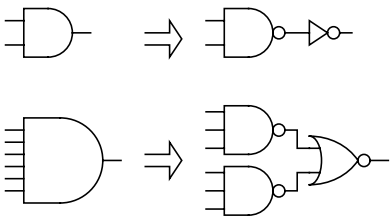


The gates shown in the slide are two input gates: you should be able to extend the principle to (e.g.) three input NAND and NOR structures simply by extending the series and parallel structures.

In all these structures each input feeds a complementary pair of transistors.

Other gate structures

'More complicated' gates may be made up of smaller CMOS gates, such as these AND gates:



It is also possible to make **complex gates** by extending the serial/parallel CMOS structures … but we won't do that here.

The other 'gates' which occur commonly are state-holding elements, such as D-type flip-flops. Although these are often pictured as being made from NAND gates 'real' implementations are typically more highly optimised 'clever' circuits.

- It is a set of elements like these which usually form the basic 'building blocks' of an ASIC design.
- Many *logical* functions come in a range of different 'drive strengths': a greater 'drive' means a reduced output impedance which, in turn, implies bigger transistors.

Transistor implementation

Each transistor is built on the chip's surface:

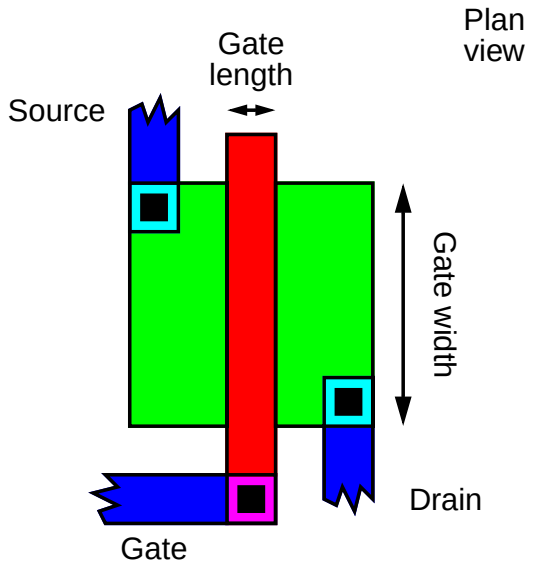
The gate 'length' is:

the distance 'through' the gate from source to drain

proportional to the transistors impedance

made as small as possible

the 'geometry' measure associated with the manufacturing process



The gate 'width' is:

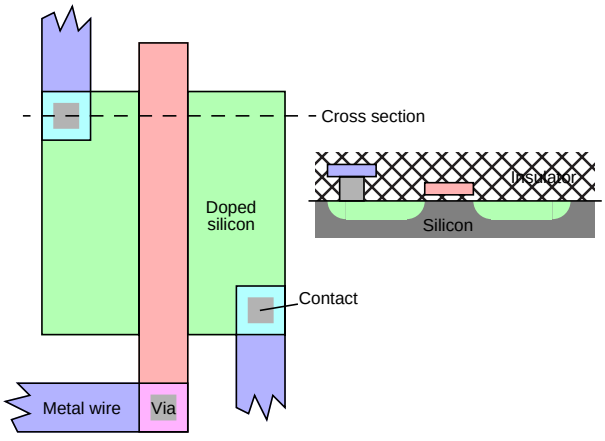
the distance 'across' the gate

inversely proportional to the transistors impedance

sized by the user (subject to a minimum)

The gate's capacitive 'load' is:

proportional to its area (length x width)



The most desirable properties of most transistors are:

- ❏ low input load
- ❏ low output impedance

Thus gate **length** should be as **small** as possible.

The gate **width** is a **compromise** between input load and output impedance.

Gate width determines the 'drive strength' of a transistor – and hence the logic gate it is used in.

Usually each logic gate will be available in a number of different 'drive strengths'.

When building a CMOS logic gate it should be apparent that some transistors are in series and therefore their effective gate lengths are added (the electric current has to flow through them all). To compensate for this it is usual to increase the transistor widths proportionately. Thus, if two transistors are in a series 'stack' their widths will be double the 'standard' size; if there are three then the width is tripled, etc.

If this is not done the gate will switch more slowly in the appropriate direction.

Another way to keep size/impedance under control is the limiting of the number of series transistors in a single logic gate. This limit is typically three or four which means than gates with more inputs are not (directly) possible. Sometimes larger gates will exist in a library but they are constructed from several smaller gates.

The effective impedance of transistors in parallel is reduced. However these transistors are not usually reduced in width because there is no guarantee that more than one will be 'on' when the gate switches.

A practical note

Although CMOS looks symmetric 'on paper' silicon implementations have to deal with real materials. The channel impedance for a PMOS transistor is higher than that in an NMOS transistor of the same dimensions. It is therefore usual to make PMOS transistors wider (typically about twice the size) of their NMOS equivalents.

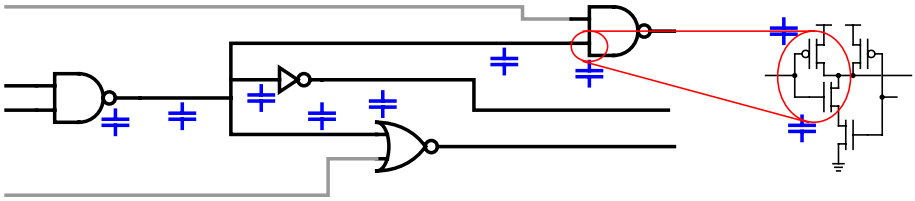
Because the input load (capacitance) of the transistor is its gate area it is desirable to minimise the total gate area loading a signal.

From the information on this page you should be able to deduce why, in general, CMOS NAND gate logic is preferable to NOR gate logic.

Loading

The most important loading on a gate is the capacitance, comprising the sum of:

- internal capacitance of the driving gate
- all the gate inputs it connects to (its '**fanout**')
- the interconnection wires



On modern processes the **wiring tends to dominate** the load.

This capacitance needs to be charged/discharged when the signal switches

- more load slows this down
- higher drive speeds this up
  - but increases the input load of the driving gate

A compromise solution is generally sought.

Edge speeds

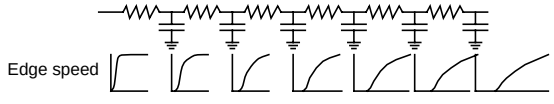
When two gates in series are switched, the first gate's output needs to reach the threshold of the second *gate* before it starts to switch. The **edge speed** of the signal is therefore important. In CMOS circuits the edge speeds of the signals along wires usually contributes the largest proportion of the delay (larger than the time for the transistors to 'switch'). For fast circuits it is therefore beneficial to provide fast edges.

There are (arguably) three ways of increasing edge speed:

- Reduce capacitive load
  - Tricky! Governed by design and physical layout
- Increase drive strength (reduce output impedance)
  - Using a 'bigger' gate is possible but will increase load on its inputs; may also occupy more space
- Cheat, by *buffering* the signal
  - Inserting an intermediate gate will mean two edges switch successively *but may still be faster*
  - CAD assistance can do this automatically

More complications

It is not just the transistors which have impedance, wires do as well. For most of VLSI history this has been sufficiently small as to be negligible in all but the longest tracks. With ever-shrinking geometry the wires get shorter (Good) but also thinner side-to-side (Bad) and top-to-bottom (also Bad). [The shrinking spacing also helps increase the capacitance.] Thus wires of any great length[1] need to be modelled as a series of 'RC' sections and the edges are slower progressively along its length.



1. A 'long' wire is of the order of 0.1 mm, maybe less.

Why slow edges are Bad (more advanced stuff)

Power

Let's use an inverter model for simplicity, and make the input low. The NMOS transistor is 'off' and the PMOS transistor is 'on', thus the output is high.

When the input moves from low to high, at some point it crosses the NMOS threshold (measured from Vss) and that transistor 'starts to conduct'.

When the input moves from low to high, at some point it crosses the PMOS threshold (measured from Vdd) and that transistor 'stops conducting'.

These may not be at the same point. During the transition both transistors may be 'on' at the same time if (when) this occurs some current flows directly from Vdd to Vss and this is wasted. A fast edge will reduce any time spent in this state and thus reduce the energy wasted (and consequential heating).

Another ploy to reduce this is to build transistors with high thresholds so that any overlap is reduced or eliminated. This works. However the transistors – and hence the gate – switching is delayed until later in the input transition so the gate delay is increased.

It is not uncommon to provide different threshold options on the same VLSI process, so a gate may be 'normal', 'high-speed', 'low-power', … Depending on requirements it may be that gates in the critical path are high-speed and gates where there is plenty of time may be low-power; CAD assistance should be available.

Noise

No system is perfect and it is likely that there will be some noise (unwanted, unpredictable variation) on every signal. In digital circuits this rarely endangers operation because the '0's and '1's are close to the power rails.

During switching noise can make a signal less certain and even make it 'change' multiple times. This is a Bad Thing – especially for power – and the threat can be minimised by switching fast rather than leaving a signal in an intermediate state for some time.

Why bother with all this?

- CAD tools hide much of the electrical detail from the logic designer
  - Both a good and a bad thing
  - Sometimes need to understand – and influence – what happens 'beneath'
  - Be aware that design apparently 'slows down' throughout the construction process
- From a circuit design it is apparent what (logical) gates are loading each node.
  - A fanout count helps to allow an *estimate* of the wiring load
  - Hence estimate edge speeds (which dominate delays)
- CAD will select gate sizes or insert buffers according to modelled logic speeds
  - But designer will specify parameters for *how* something is optimised
  - May be able to optimise for speed or power or area or …

More on CAD tools later.

Effect of layout

A logical design which is reduced to a set of interconnected gates is usually referred to as a '**netlist**'.

Unless the tools have malfunctioned (which is unlikely, but it's usual to go back and check anyway because software sometimes does have bugs!) this implements the same logical function as the designer wrote.

This netlist can be simulated. An advantage here is that, given a gate implementation, the likely critical path(s) are more evident and the tools can start to 'back annotate' gate delays into the simulation. This may be the first 'real'[1] indication of how fast the circuit may go. This is called '**pre-layout**' simulation. This will be slower than a behavioural simulation because, for example, an adder is not now an 'adder' (possibly with an estimated delay) but a collection of gates, each of which is modelled separately.

Once the gates are laid-out on the chip a more accurate estimate of the wiring capacitance may be '**extracted**'. This allows '**post-layout**' simulation which will give a better performance estimate – almost inevitably somewhat slower than the pre-layout simulation so be prepared for disappointment!

Post-layout simulation may include wiring broken into short sections to increase the accuracy further; of course this means even more components have to be modelled and the simulation times increase commensurately.

1. An experienced designer will have *some idea* of the logic depth resulting in HDL code, just as an experienced programmer will have an idea of the efficiency of the compiled machine code.

'Mixed Signal' ASICs

'Mixed signal' is the term used to refer to devices which incorporate both digital and analogue components. Analogue parts are still hand-designed and imported as macrocells[2].

Analogue design is a separate skill – this is 'real' electronics – and is not covered here. However electronic equipment often contains some analogue circuitry and integrating this onto a single SoC makes economic sense. Therefore modern SoCs typically contain some analogue circuits.

ASIC processes are designed for digital circuits and digital circuits are more tolerant of process variation. Therefore a good rule is to try to minimise the analogue content of a device.

Examples of analogue circuitry typically used in 'digital' devices include:

- radio transceivers
- clock multipliers
- temperature sensors
- …

2. Covered in a subsequent lecture.