- Concept:
  1. micro-electronic devices are based on micro-electronic **switches**:

$$\begin{array}{rcrcl} \text{switch is } \textit{on} & \Rightarrow & \text{connected} & \rightsquigarrow & \text{allows flow of electrons} \\ \text{switch is } \textit{off} & \Rightarrow & \text{disconnected} & \rightsquigarrow & \text{disallows flow of electrons} \end{array}$$

  2. however, we need those switches to be
     - efficient in terms of space, i.e., their physical size
     - efficient in terms of time, i.e., how quickly they operate
     - reliable, i.e., low probability of failure
     - manufacturable, e.g., high yield, even given high complexity and density
     - useable, i.e., packaged into high(er)-level building blocks

- Agenda:

$$\textbf{semi-conductors} \rightsquigarrow \textbf{transistors} \rightsquigarrow \textbf{logic gates},$$

  i.e.,

  1. physical foundations,
  2. semi-conductors and transistors,
  3. logic gates, and
  4. physical constraints and properties.

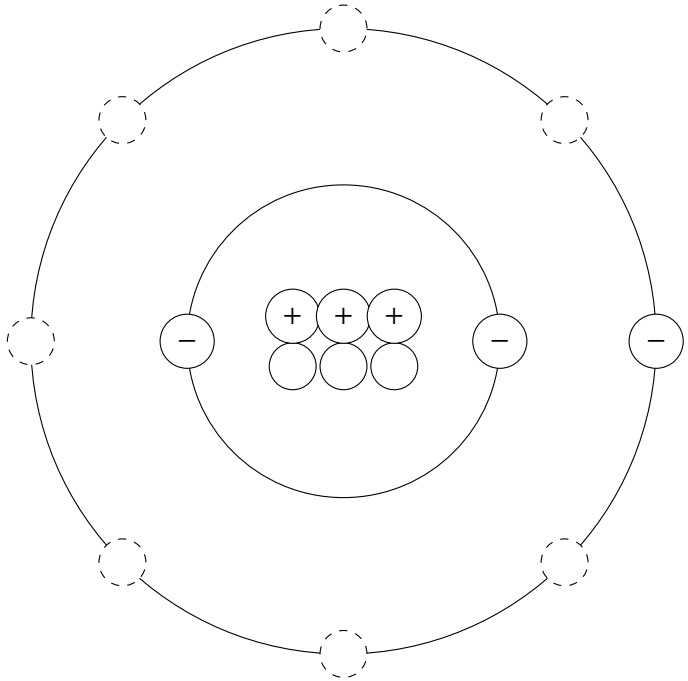
- Caveat!
  - this is CS not EE, so we carefully limit the level of detail,
  - e.g., "transistors are just switches" *is* reasonable (although clearly there's more to it than that).

Part 1: foundations (1)

https://xkcd.com/567

- Fact #1:
  - A given **atom** is built from
    1. a group of nucleons, either **protons** or **neutrons**, called the nucleus, and
    2. a cloud of **electrons** arranged in **shells**,

    e.g., lithium (Li):



- The number of protons dictates **atomic number**; the number of neutrons dictates **isotope**.
- The electron configuration is not arbitrary: the n-th shell can accommodate upto $2n^2$ electrons.
- An unfilled slot within a shell is called a **hole**.

- Fact #2:
  - Each sub-atomic particle has an associated electrical **charge**, i.e.,

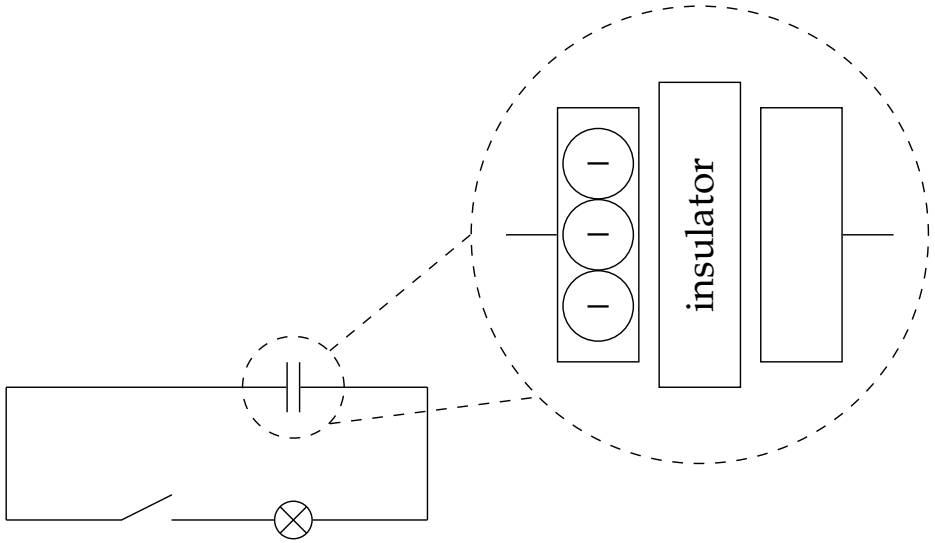
$$\begin{array}{rcrl} \text{electrons} & \mapsto & \text{negative charge} & (\text{hence } -) \\ \text{protons} & \mapsto & \text{positive charge} & (\text{hence } +) \\ \text{neutrons} & \mapsto & \text{neutral charge} & \end{array}$$

  and an **ion** is an atom with a non-zero overall charge.

- Fact #2:

  - The binding between particles can be disrupted:
    - As an electron absorbs more energy, it becomes excited; at some threshold, it will be displaced and then becomes **free**.
    - A free electron can move between shells, *or* between atoms (i.e., between their outer shells); roughly speaking, they are "attracted" by holes.

- Fact #3:
  - An electrical **current** is a flow of electrons, i.e., a flow of charge.
  - Free electrons (bound to atoms or not) "move" from low to high potential: this is how



i.e., a **capacitor** (or **battery**) works.

- Fact #3:

- A given material can be classified in terms of how easily electrons can move:

conductor (e.g., a metal) $\mapsto$ high-conductivity $\equiv$ low-resistivity
$\mapsto$ does allow electrons to move easily

insulator (e.g., a vacuum) $\mapsto$ low-conductivity $\equiv$ high-resistivity
$\mapsto$ does not allow electrons to move easily

- Fact #4:
  - Silicon (Si) is really useful, because
    1. it's **abundant**, i.e., there's lots of it and so it doesn't cost much,
    2. it's fairly **inert**, i.e., it's stable enough not to react in weird ways with other things, *and*
    3. it can be **doped** with a donor material, e.g.,

silicon (Si)	+	(	boron (B)	or	aluminium (Al)	)	$\rightsquigarrow$	extra holes
silicon (Si)	+	(	phosphorus (P)	or	arsenic (As)	)	$\rightsquigarrow$	extra electrons

    allowing construction of materials with specific sub-atomic properties.

- Fact #4:

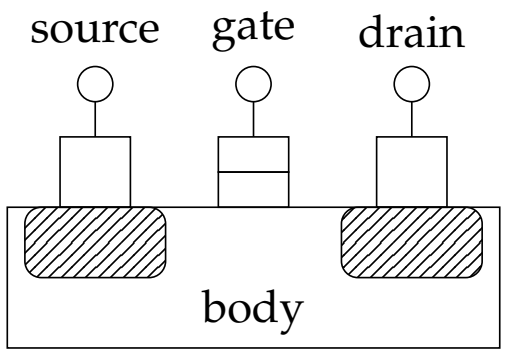
- The result is a ***semi*-conductor**:

$$\begin{array}{lcl} \text{extra holes} & \rightsquigarrow & \text{P-type semi-conductor} \\ \text{extra electrons} & \rightsquigarrow & \text{N-type semi-conductor} \end{array}$$

- A "sandwich" of P-type and N-type layers means electrons can only move *one* direction, e.g., from an N-type layer to a P-type layer, but *not* vice versa.

- Concept: a **Metal Oxide Silicon Field Effect Transistor (MOSFET)**

Circuit



where

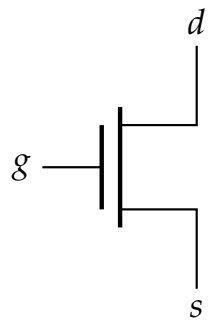
- FET transistors allow charge to flow through a conductive channel between source and drain terminals,
- the channel width, and hence conductivity, is controlled by the potential difference applied to gate terminal,
- in a MOSFET transistor, the channel is *induced* (versus a JFET, where an explicit semi-conductor layer is used).

Definition

An **N-MOSFET** (or **N-type MOSFET**, or **N-channel MOSFET**, or **NPN MOSFET**) is constructed from N-type semiconductor terminals and a P-type body:

- applying a potential difference to the gate widens the conductive channel, meaning source and drain are connected (i.e., act like a conductor); the transistor is activated.
- removing the potential difference from the gate narrows the conductive channel, meaning source and drain are disconnected (i.e., act like an insulator); the transistor is deactivated.

Using d, s and g to denote the drain, source and gate terminals, an N-MOSFET is described symbolically as

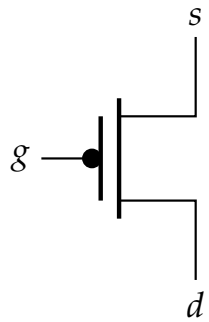


Definition

A **P-MOSFET** (or **P-type MOSFET**, or **P-channel MOSFET**, or **PNP MOSFET**) is constructed from P-type semiconductor terminals and an N-type body:

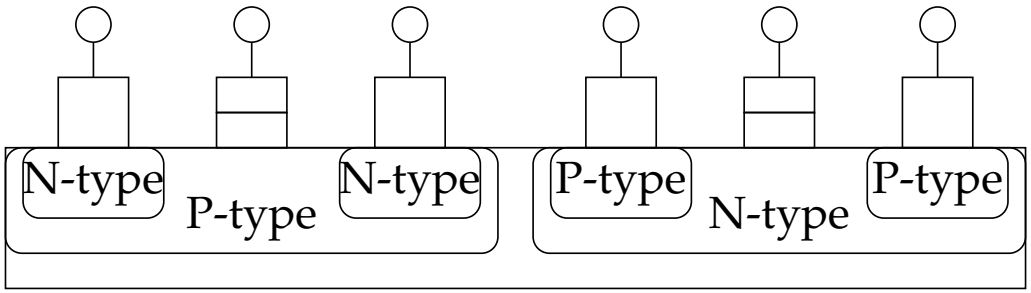
- applying a potential difference to the gate narrows the conductive channel, meaning source and drain are disconnected (i.e., act like an insulator); the transistor is deactivated.
- removing the potential difference from the gate widens the conductive channel, meaning source and drain are connected (i.e., act like a conductor); the transistor is activated.

Using d, s and g to denote the drain, source and gate terminals, an P-MOSFET is described symbolically as



- Concept: a **Complementary Metal-Oxide-Semiconductor (CMOS)** *cell*

Circuit



where

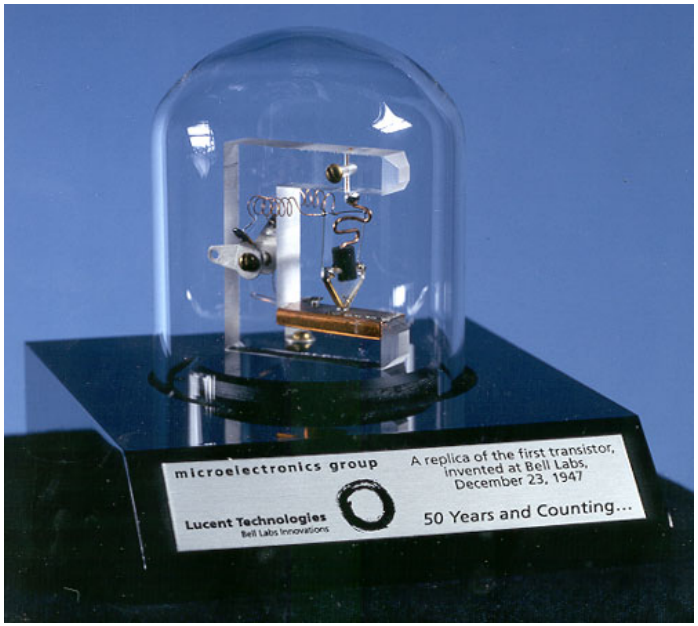
1. each cell combines one N-MOSFET and one P-MOSFET,
2. only switching from one state to another consumes much power (or **dynamic consumption**) since one transistor does the opposite of the other,
3. there isn't much "leakage" (or **static consumption**) at other times.

An Aside: some history

- A historically dominant switch technology was the **vacuum tube**:
  - Looks like a light-bulb with a glass envelope holding a vacuum.
  - Inside vacuum is an electron producing filament (or cathode) and a metal plate (or anode).
  - When filament is heated, electrons are produced into vacuum which are attracted by plate.
  - Reliability of vacuum tubes was reasonably good, but most failed during power-on or power-off ...
  - ... the terms **bug** and **debug** both allegedly stem (in part) from failure of this sort.

https://en.wikipedia.org/wiki/File:6P1P.jpg

An Aside: some history



- The replacement for this generation of technology is the **transistor**.
- There are *many* types, but a potted overview of the one we'll focus on is:
  - 1925: **Field Effect Transistor (FET)**.
  - 1952: **junction gate FET (or JFET)**.
  - 1960: **Metal Oxide Semi-conductor FET (MOSFET)**.

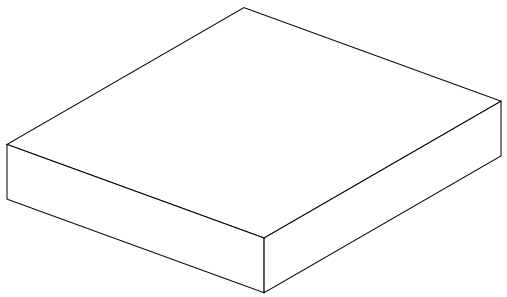
https://en.wikipedia.org/wiki/File:Replica-of-first-transistor.jpg

- Concept: **Complementary Metal-Oxide-Semiconductor (CMOS)** *fabrication*

Algorithm

1. Start with a clean, prepared **wafer**.
2. Apply a layer of **substrate** material, e.g., metal or semi-conductor.
3. Apply a layer of **photoresist** material.
4. Expose the photoresist to a precise negative or **mask** of design; this hardens the exposed photoresist.
5. Wash away unhardened photoresist.
6. Etch away uncovered substrate.
7. Strip hardened photoresist.

Example



where

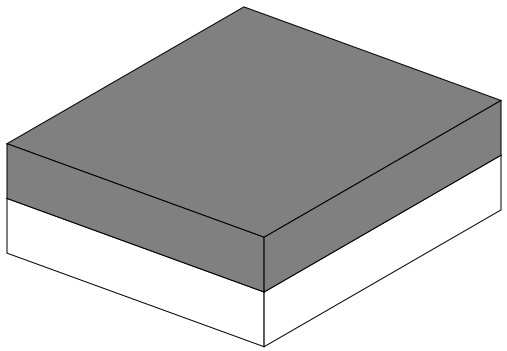
- the algorithm iterates to produce many layers, i.e., the result is 3D not 2D,
- regularity offers a significant advantage: we can manufacture *many* similar components in a layer within one iteration,
- the feature size (e.g., 90nm CMOS), and so density, relates to the resolution of this process.

- Concept: **Complementary Metal-Oxide-Semiconductor (CMOS)** *fabrication*

Algorithm

1. Start with a clean, prepared **wafer**.
2. Apply a layer of **substrate** material, e.g., metal or semi-conductor.
3. Apply a layer of **photoresist** material.
4. Expose the photoresist to a precise negative or **mask** of design; this hardens the exposed photoresist.
5. Wash away unhardened photoresist.
6. Etch away uncovered substrate.
7. Strip hardened photoresist.

Example



where

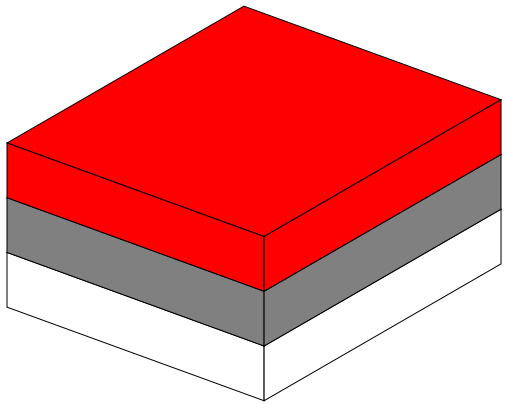
- the algorithm iterates to produce many layers, i.e., the result is 3D not 2D,
- regularity offers a significant advantage: we can manufacture *many* similar components in a layer within one iteration,
- the feature size (e.g., 90nm CMOS), and so density, relates to the resolution of this process.

- Concept: **Complementary Metal-Oxide-Semiconductor (CMOS)** *fabrication*

Algorithm

1. Start with a clean, prepared **wafer**.
2. Apply a layer of **substrate** material, e.g., metal or semi-conductor.
3. Apply a layer of **photoresist** material.
4. Expose the photoresist to a precise negative or **mask** of design; this hardens the exposed photoresist.
5. Wash away unhardened photoresist.
6. Etch away uncovered substrate.
7. Strip hardened photoresist.

Example



where

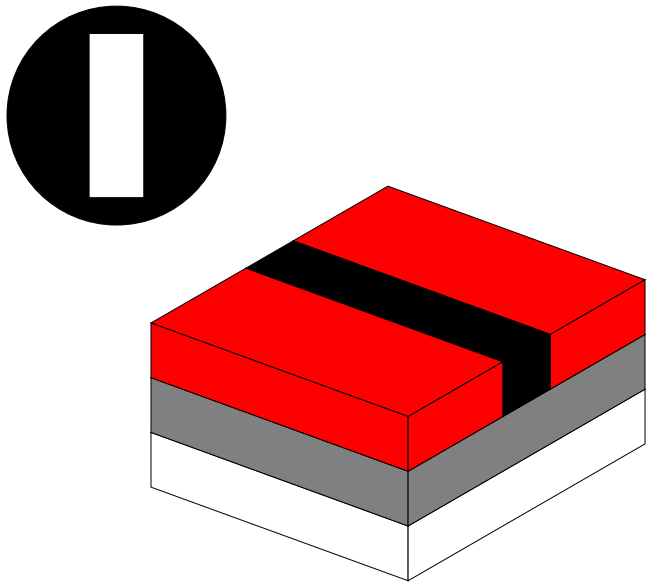
- the algorithm iterates to produce many layers, i.e., the result is 3D not 2D,
- regularity offers a significant advantage: we can manufacture *many* similar components in a layer within one iteration,
- the feature size (e.g., 90nm CMOS), and so density, relates to the resolution of this process.

- Concept: **Complementary Metal-Oxide-Semiconductor (CMOS)** *fabrication*

Algorithm

1. Start with a clean, prepared **wafer**.
2. Apply a layer of **substrate** material, e.g., metal or semi-conductor.
3. Apply a layer of **photoresist** material.
4. Expose the photoresist to a precise negative or **mask** of design; this hardens the exposed photoresist.
5. Wash away unhardened photoresist.
6. Etch away uncovered substrate.
7. Strip hardened photoresist.

Example



where

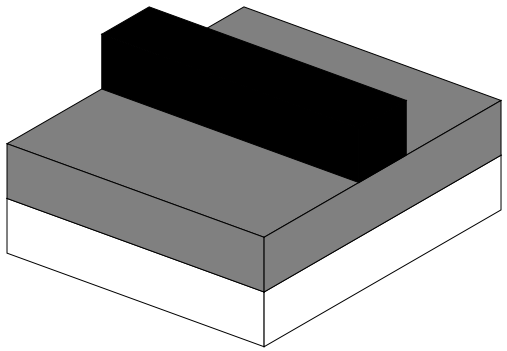
- the algorithm iterates to produce many layers, i.e., the result is 3D not 2D,
- regularity offers a significant advantage: we can manufacture *many* similar components in a layer within one iteration,
- the feature size (e.g., 90nm CMOS), and so density, relates to the resolution of this process.

- Concept: **Complementary Metal-Oxide-Semiconductor (CMOS)** *fabrication*

Algorithm

1. Start with a clean, prepared **wafer**.
2. Apply a layer of **substrate** material, e.g., metal or semi-conductor.
3. Apply a layer of **photoresist** material.
4. Expose the photoresist to a precise negative or **mask** of design; this hardens the exposed photoresist.
5. Wash away unhardened photoresist.
6. Etch away uncovered substrate.
7. Strip hardened photoresist.

Example



where

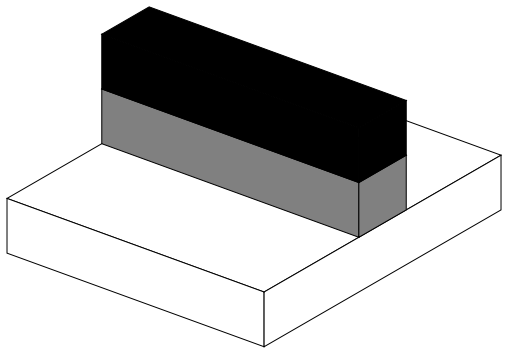
- the algorithm iterates to produce many layers, i.e., the result is 3D not 2D,
- regularity offers a significant advantage: we can manufacture *many* similar components in a layer within one iteration,
- the feature size (e.g., 90nm CMOS), and so density, relates to the resolution of this process.

- Concept: **Complementary Metal-Oxide-Semiconductor (CMOS)** *fabrication*

Algorithm

1. Start with a clean, prepared **wafer**.
2. Apply a layer of **substrate** material, e.g., metal or semi-conductor.
3. Apply a layer of **photoresist** material.
4. Expose the photoresist to a precise negative or **mask** of design; this hardens the exposed photoresist.
5. Wash away unhardened photoresist.
6. Etch away uncovered substrate.
7. Strip hardened photoresist.

Example



where

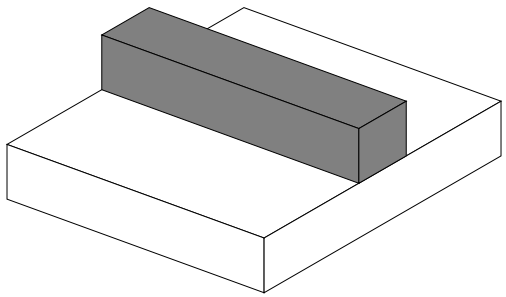
- the algorithm iterates to produce many layers, i.e., the result is 3D not 2D,
- regularity offers a significant advantage: we can manufacture *many* similar components in a layer within one iteration,
- the feature size (e.g., 90nm CMOS), and so density, relates to the resolution of this process.

- Concept: **Complementary Metal-Oxide-Semiconductor (CMOS)** *fabrication*

Algorithm

1. Start with a clean, prepared **wafer**.
2. Apply a layer of **substrate** material, e.g., metal or semi-conductor.
3. Apply a layer of **photoresist** material.
4. Expose the photoresist to a precise negative or **mask** of design; this hardens the exposed photoresist.
5. Wash away unhardened photoresist.
6. Etch away uncovered substrate.
7. Strip hardened photoresist.

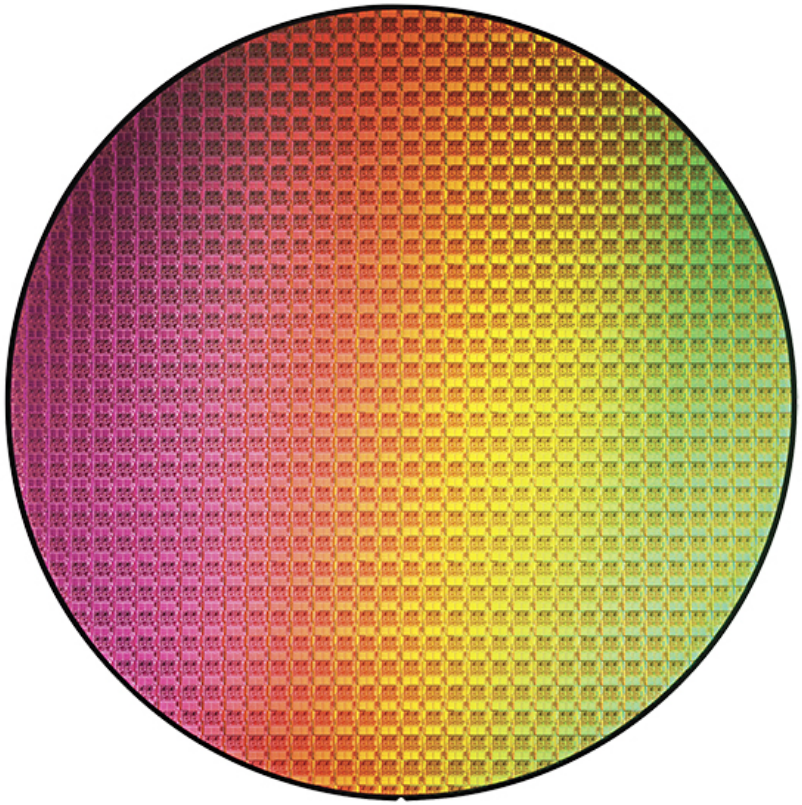
Example

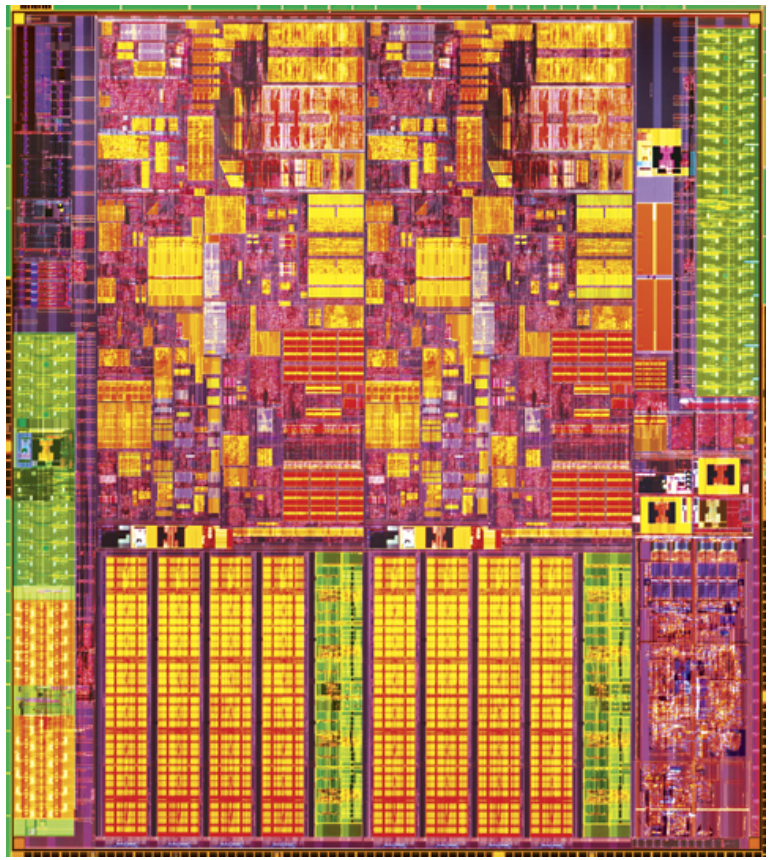


where

- the algorithm iterates to produce many layers, i.e., the result is 3D not 2D,
- regularity offers a significant advantage: we can manufacture *many* similar components in a layer within one iteration,
- the feature size (e.g., 90nm CMOS), and so density, relates to the resolution of this process.

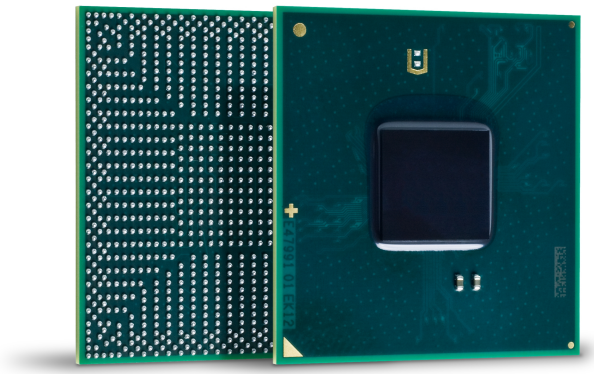
- Concept: the result is a **wafer**, e.g.,





https://www.intel.com/pressroom/archive/releases/2010/20100107comp_sm.htm

- Concept: each component is **packaged** before use, e.g.,



which

- protects against damage, often including a heat sink as well, *and*
- provides a usable interface, via external **pins** bonded to internal inputs and outputs.

https://www.intel.com/pressroom/archive/releases/2010/20100107comp_sm.htm

Moore's Law: originally an observation

> *The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least* 10 *years. That means by* 1975*, the number of components per integrated circuit for minimum cost will be* 65,000*.*
>
> *– Moore [8]*

and later updated: in short "number of transistors in unit area doubles roughly every two years".

https://download.intel.com/pressroom/images/events/moores_law_40th/Gordon_Moore/GordonMoore_young.jpg

Example (Moore's Law [8], from 1970 to 2005)

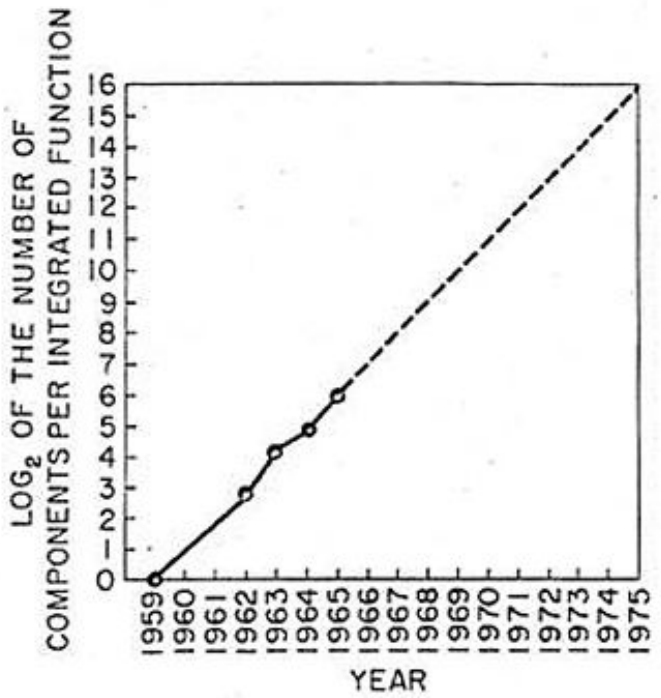
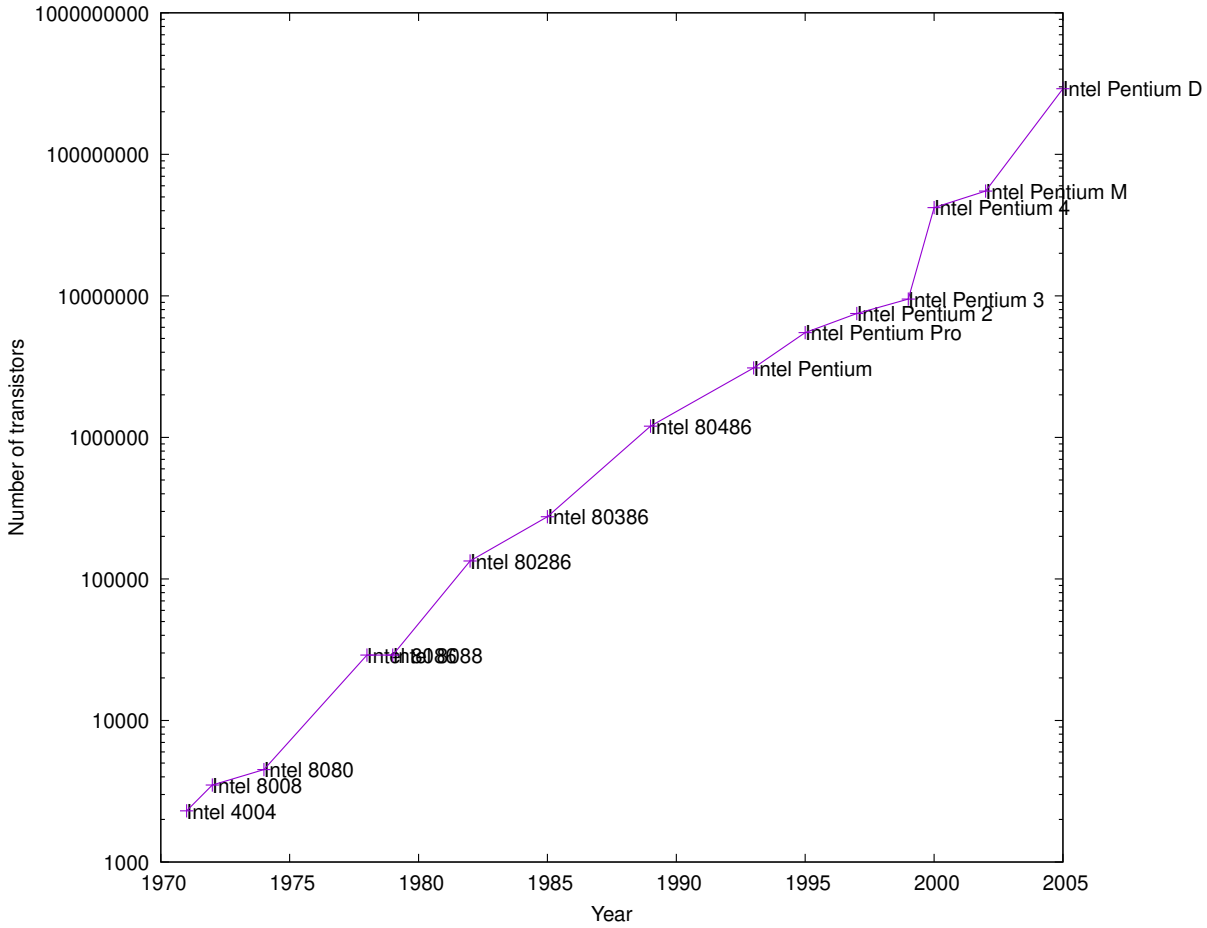
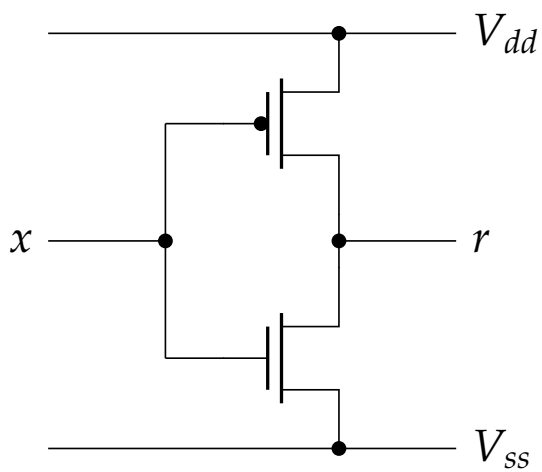


Fig. 2 Number of components per integrated function for minimum cost per component extrapolated vs time.

Example (Moore's Law [8], from 1970 to 2005)

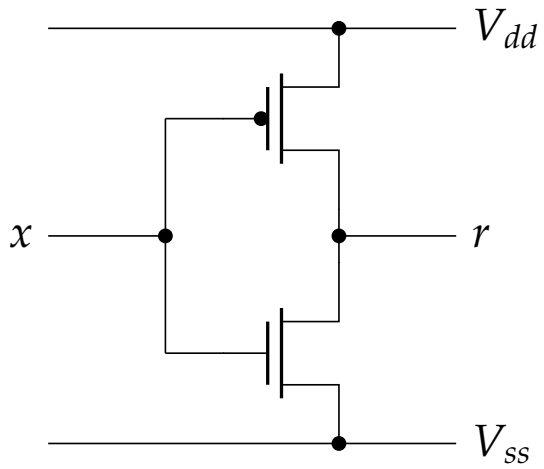


Circuit



- Question: what does this organisation of MOSFET transistors *do*?

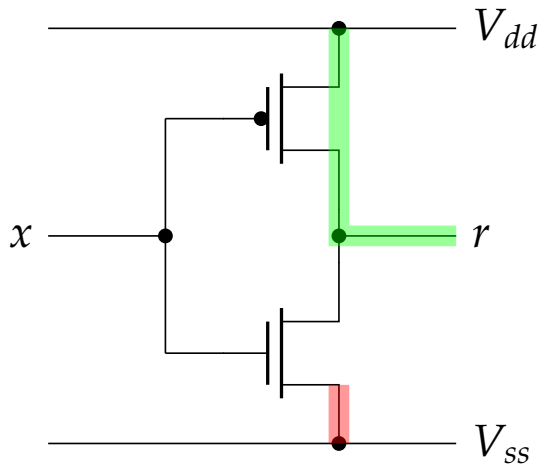
Circuit



- Connect x to V_{ss}:
  1. the top P-MOSFET will be connected,
  2. the bottom N-MOSFET will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{dd}:
  1. the top P-MOSFET will be disconnected,
  2. the bottom N-MOSFET will be connected,
  3. r will be connected to V_{ss}.

- Question: what does this organisation of MOSFET transistors *do*?
- Answer: this matches the behaviour of NOT, i.e., it's an **inverter**!

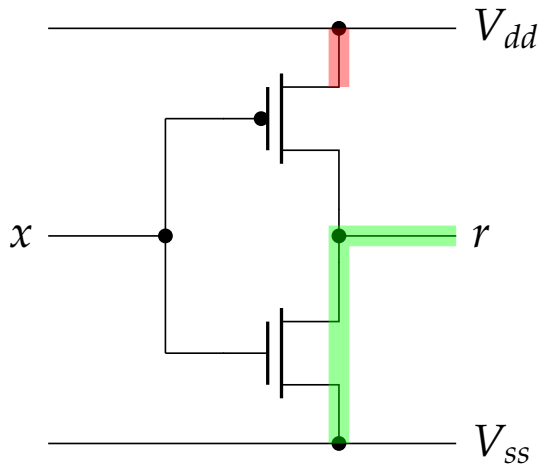
Circuit



- Connect x to V_{ss}:
  1. the top P-MOSFET will be connected,
  2. the bottom N-MOSFET will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{dd}:
  1. the top P-MOSFET will be disconnected,
  2. the bottom N-MOSFET will be connected,
  3. r will be connected to V_{ss}.

- Question: what does this organisation of MOSFET transistors *do*?
- Answer: this matches the behaviour of NOT, i.e., it's an **inverter**!

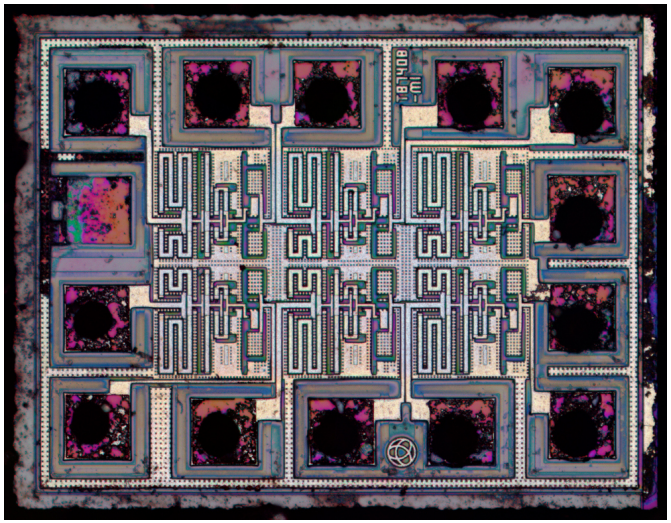
Circuit



- Connect x to V_{ss}:
  1. the top P-MOSFET will be connected,
  2. the bottom N-MOSFET will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{dd}:
  1. the top P-MOSFET will be disconnected,
  2. the bottom N-MOSFET will be connected,
  3. r will be connected to V_{ss}.

- Question: what does this organisation of MOSFET transistors *do*?
- Answer: this matches the behaviour of NOT, i.e., it's an **inverter**!

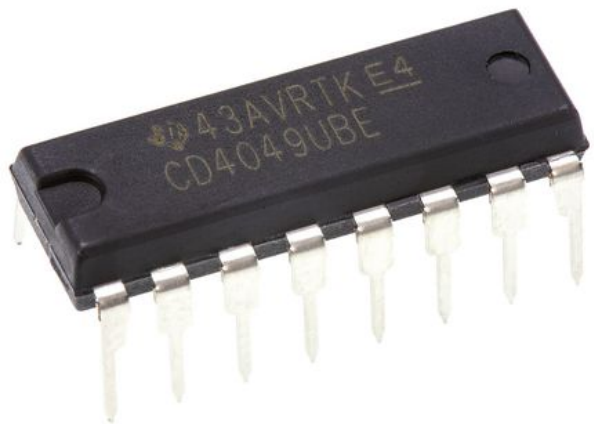
Circuit



- Question: what does this organisation of MOSFET transistors *do*?
- Answer: this matches the behaviour of NOT, i.e., it's an **inverter**!

https://zeptobars.com/en/read/CD4049-cmos-inverter-metal-gate

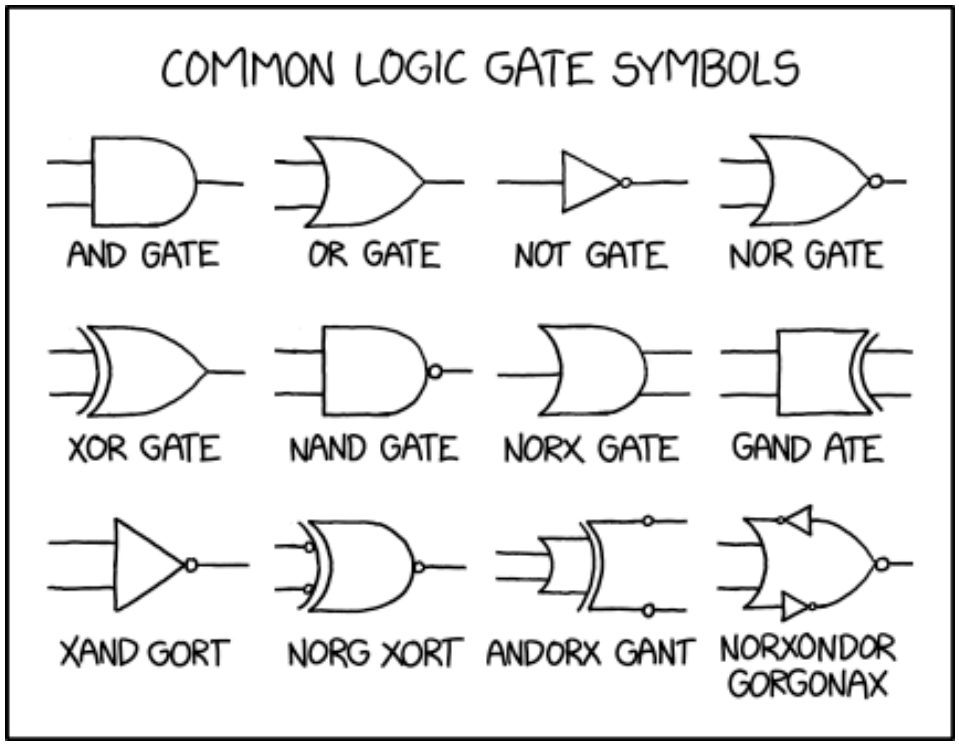
Circuit



- Question: what does this organisation of MOSFET transistors *do*?
- Answer: this matches the behaviour of NOT, i.e., it's an **inverter**!

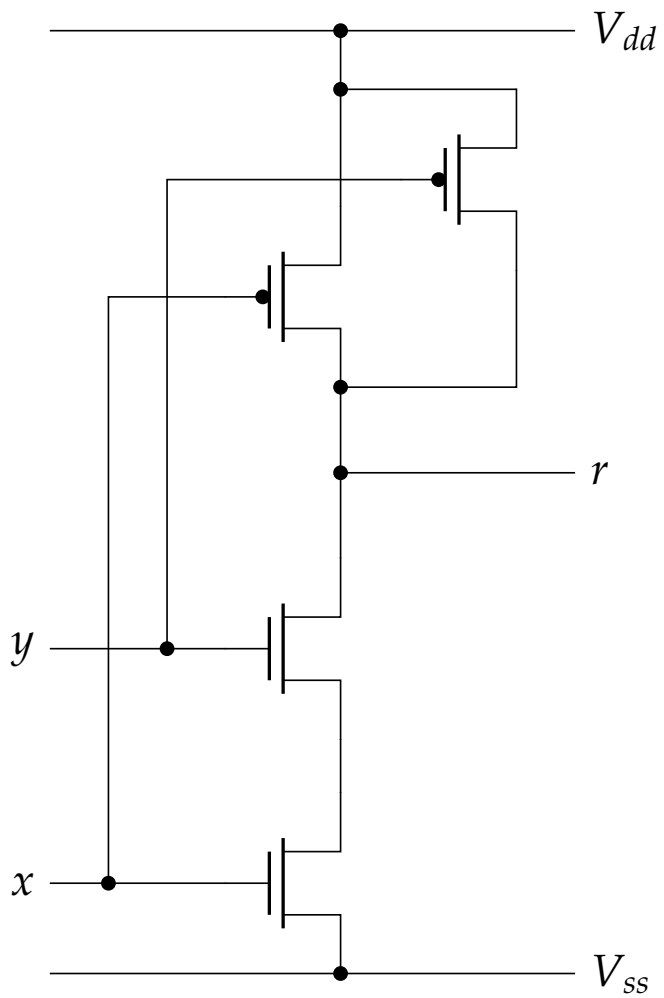
https://www.ti.com/product/CD4049UB

Part 3: logic gates (2)



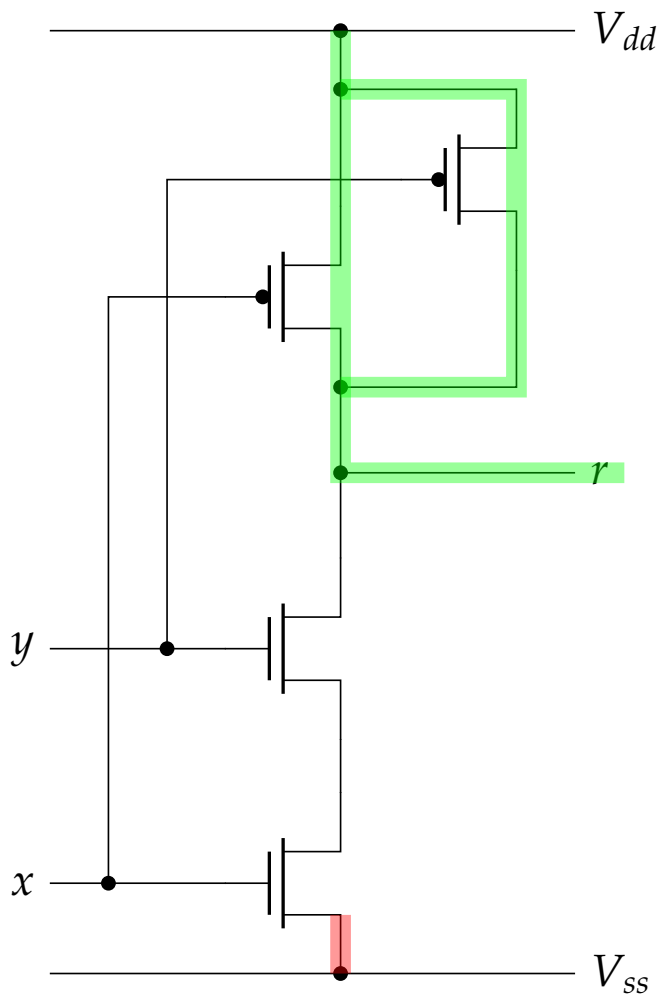
https://xkcd.com/2497

Circuit



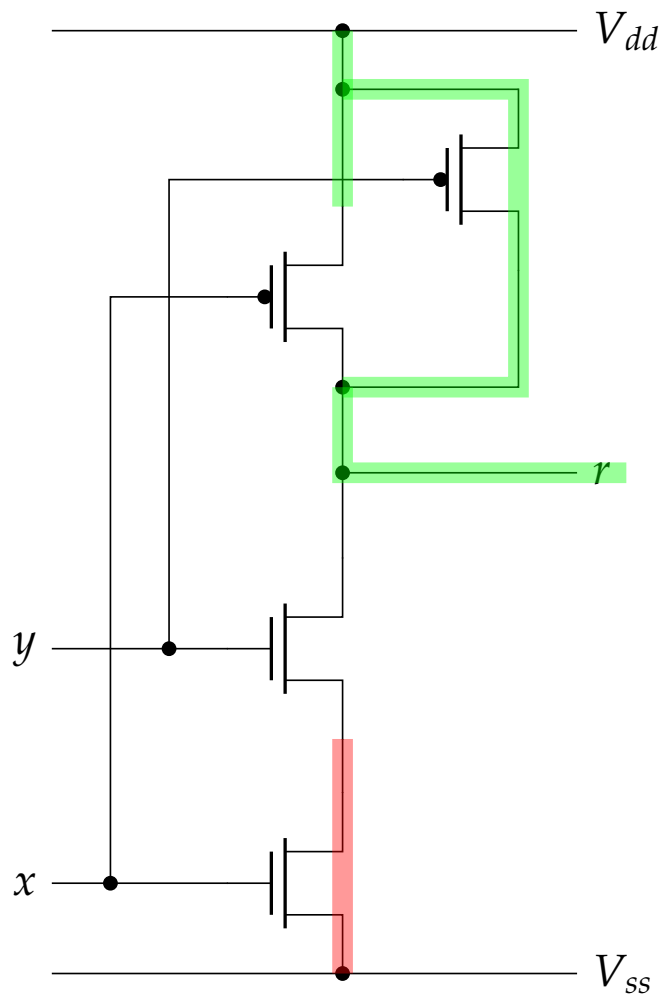
- Connect both x and y to V_{ss}:
  1. both top P-MOSFETs will be connected,
  2. both bottom N-MOSFETS will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{dd} and y to V_{ss}:
  1. the right-most P-MOSFET will be connected,
  2. the upper-most N-MOSFET will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{ss} and y to V_{dd}:
  1. the left-most P-MOSFET will be connected,
  2. the lower-most N-MOSFET will be disconnected,
  3. r will be connected to V_{dd}.
- Connect both x and y to V_{dd}:
  1. both top P-MOSFETs will be disconnected,
  2. both bottom N-MOSFETS will be connected,
  3. r will be connected to V_{ss}.

Circuit



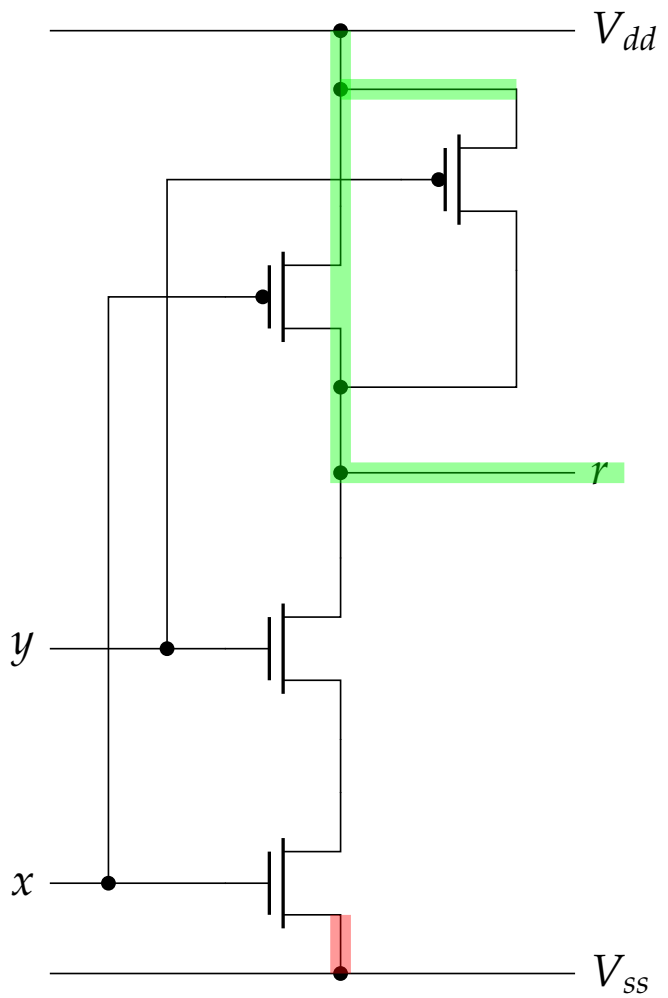
- Connect both x and y to V_{ss}:
  1. both top P-MOSFETs will be connected,
  2. both bottom N-MOSFETS will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{dd} and y to V_{ss}:
  1. the right-most P-MOSFET will be connected,
  2. the upper-most N-MOSFET will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{ss} and y to V_{dd}:
  1. the left-most P-MOSFET will be connected,
  2. the lower-most N-MOSFET will be disconnected,
  3. r will be connected to V_{dd}.
- Connect both x and y to V_{dd}:
  1. both top P-MOSFETs will be disconnected,
  2. both bottom N-MOSFETS will be connected,
  3. r will be connected to V_{ss}.

Circuit



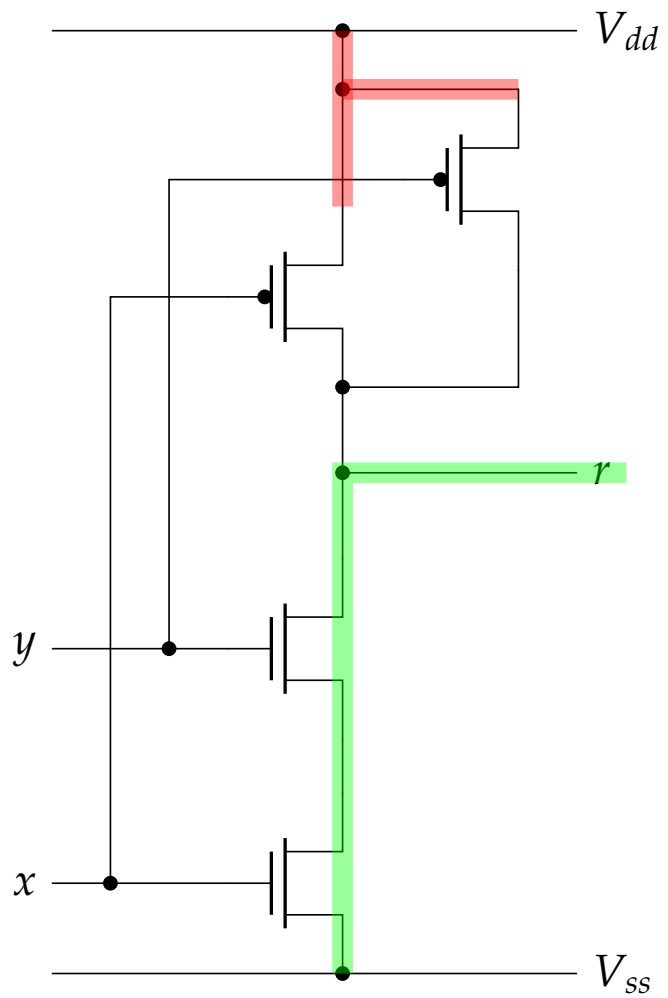
- Connect both x and y to V_{ss}:
  1. both top P-MOSFETs will be connected,
  2. both bottom N-MOSFETS will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{dd} and y to V_{ss}:
  1. the right-most P-MOSFET will be connected,
  2. the upper-most N-MOSFET will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{ss} and y to V_{dd}:
  1. the left-most P-MOSFET will be connected,
  2. the lower-most N-MOSFET will be disconnected,
  3. r will be connected to V_{dd}.
- Connect both x and y to V_{dd}:
  1. both top P-MOSFETs will be disconnected,
  2. both bottom N-MOSFETS will be connected,
  3. r will be connected to V_{ss}.

Circuit



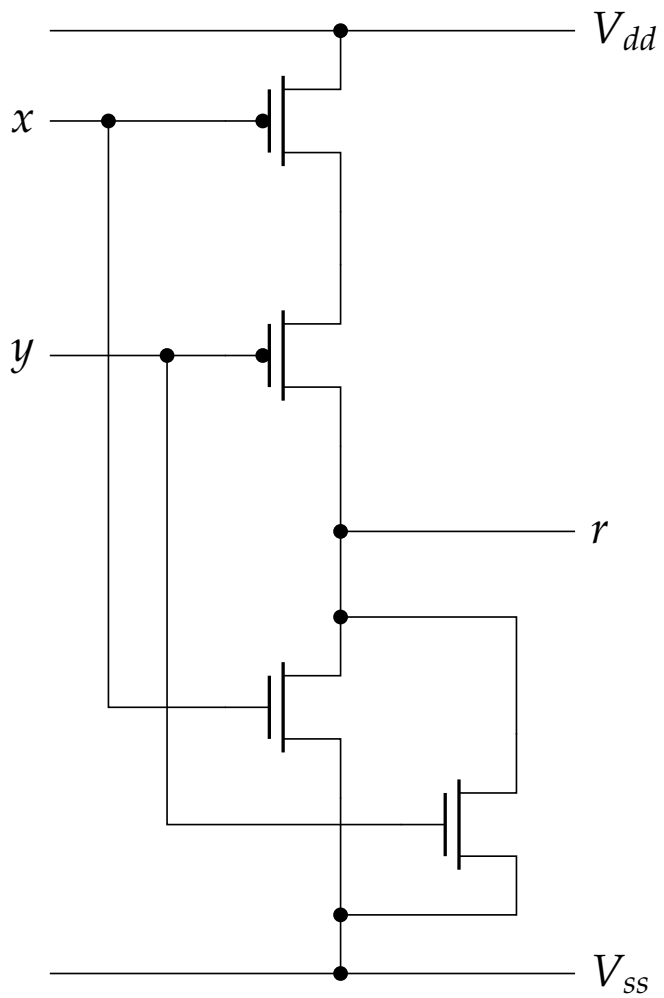
- Connect both x and y to V_{ss}:
  1. both top P-MOSFETs will be connected,
  2. both bottom N-MOSFETS will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{dd} and y to V_{ss}:
  1. the right-most P-MOSFET will be connected,
  2. the upper-most N-MOSFET will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{ss} and y to V_{dd}:
  1. the left-most P-MOSFET will be connected,
  2. the lower-most N-MOSFET will be disconnected,
  3. r will be connected to V_{dd}.
- Connect both x and y to V_{dd}:
  1. both top P-MOSFETs will be disconnected,
  2. both bottom N-MOSFETS will be connected,
  3. r will be connected to V_{ss}.

Circuit



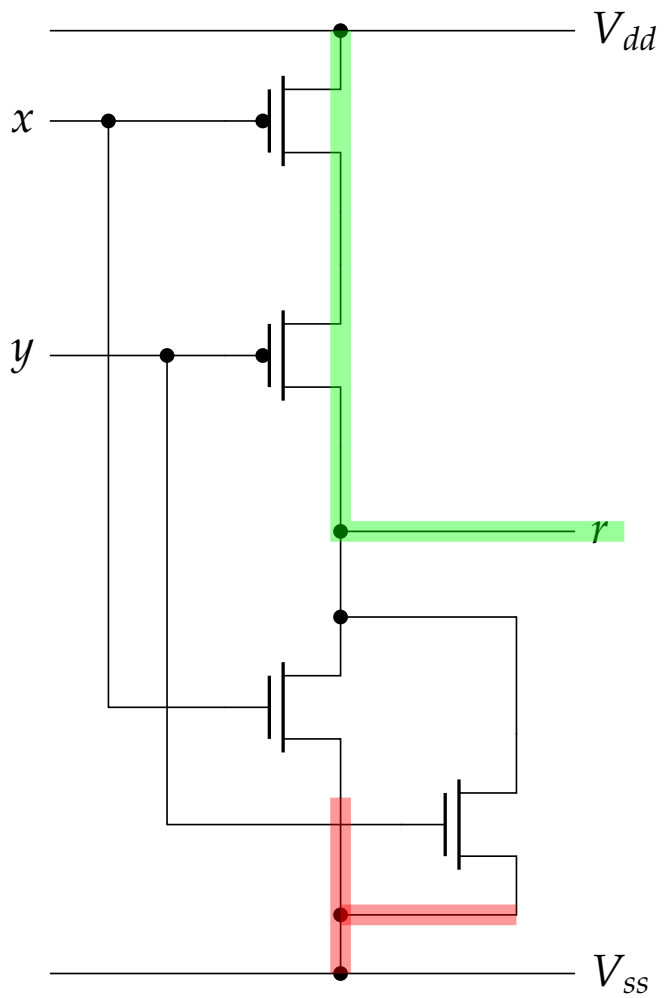
- Connect both x and y to V_{ss}:
  1. both top P-MOSFETs will be connected,
  2. both bottom N-MOSFETS will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{dd} and y to V_{ss}:
  1. the right-most P-MOSFET will be connected,
  2. the upper-most N-MOSFET will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{ss} and y to V_{dd}:
  1. the left-most P-MOSFET will be connected,
  2. the lower-most N-MOSFET will be disconnected,
  3. r will be connected to V_{dd}.
- Connect both x and y to V_{dd}:
  1. both top P-MOSFETs will be disconnected,
  2. both bottom N-MOSFETS will be connected,
  3. r will be connected to V_{ss}.

Circuit



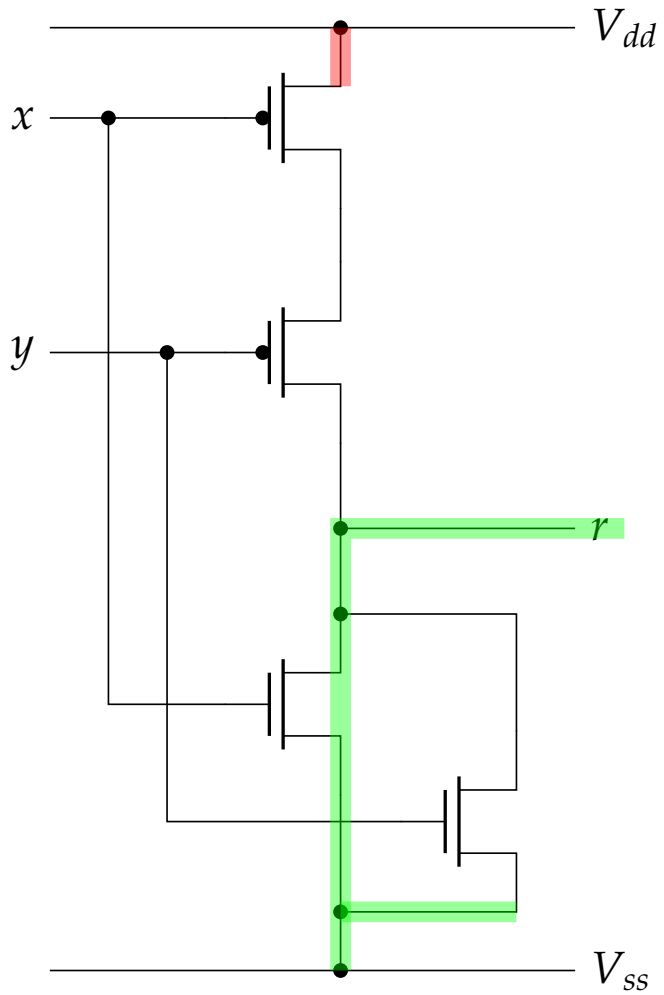
- Connect both x and y to V_{ss}:
  1. both top P-MOSFETs will be connected,
  2. both bottom N-MOSFETS will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{dd} and y to V_{ss}:
  1. the upper-most P-MOSFET will be disconnected,
  2. the left-most N-MOSFET will be connected,
  3. r will be connected to V_{ss}.
- Connect x to V_{ss} and y to V_{dd}:
  1. the lower-most P-MOSFET will be disconnected,
  2. the right-most N-MOSFET will be connected,
  3. r will be connected to V_{ss}.
- Connect both x and y to V_{dd}:
  1. both top P-MOSFETs will be disconnected,
  2. both bottom N-MOSFETS will be connected,
  3. r will be connected to V_{ss}.

Circuit



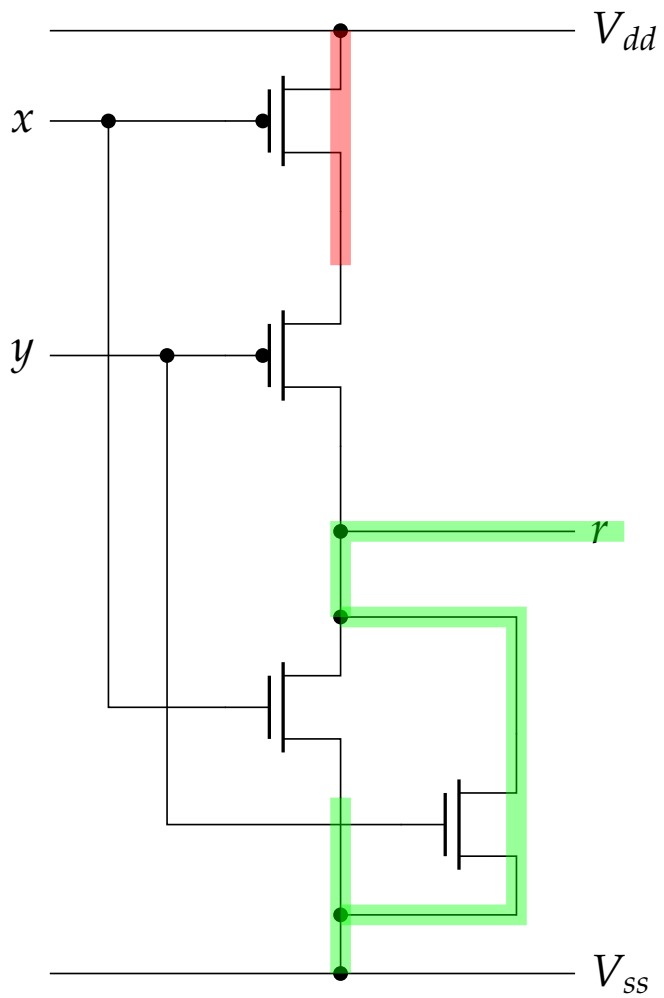
- Connect both x and y to V_{ss}:
  1. both top P-MOSFETs will be connected,
  2. both bottom N-MOSFETS will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{dd} and y to V_{ss}:
  1. the upper-most P-MOSFET will be disconnected,
  2. the left-most N-MOSFET will be connected,
  3. r will be connected to V_{ss}.
- Connect x to V_{ss} and y to V_{dd}:
  1. the lower-most P-MOSFET will be disconnected,
  2. the right-most N-MOSFET will be connected,
  3. r will be connected to V_{ss}.
- Connect both x and y to V_{dd}:
  1. both top P-MOSFETs will be disconnected,
  2. both bottom N-MOSFETS will be connected,
  3. r will be connected to V_{ss}.

Circuit



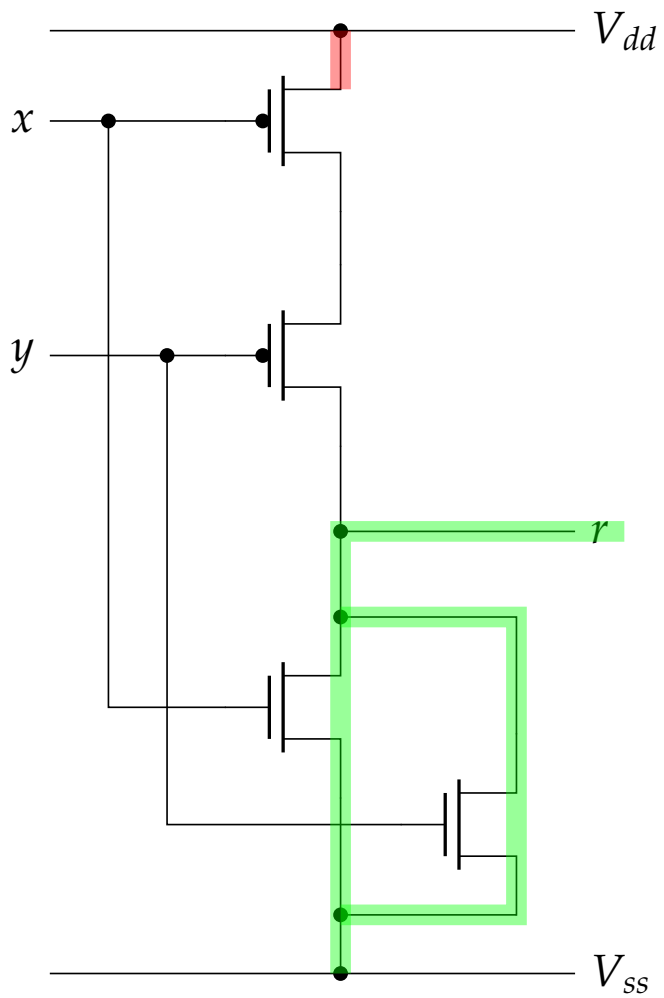
- Connect both x and y to V_{ss}:
  1. both top P-MOSFETs will be connected,
  2. both bottom N-MOSFETS will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{dd} and y to V_{ss}:
  1. the upper-most P-MOSFET will be disconnected,
  2. the left-most N-MOSFET will be connected,
  3. r will be connected to V_{ss}.
- Connect x to V_{ss} and y to V_{dd}:
  1. the lower-most P-MOSFET will be disconnected,
  2. the right-most N-MOSFET will be connected,
  3. r will be connected to V_{ss}.
- Connect both x and y to V_{dd}:
  1. both top P-MOSFETs will be disconnected,
  2. both bottom N-MOSFETS will be connected,
  3. r will be connected to V_{ss}.

Circuit



- Connect both x and y to V_{ss}:
  1. both top P-MOSFETs will be connected,
  2. both bottom N-MOSFETS will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{dd} and y to V_{ss}:
  1. the upper-most P-MOSFET will be disconnected,
  2. the left-most N-MOSFET will be connected,
  3. r will be connected to V_{ss}.
- Connect x to V_{ss} and y to V_{dd}:
  1. the lower-most P-MOSFET will be disconnected,
  2. the right-most N-MOSFET will be connected,
  3. r will be connected to V_{ss}.
- Connect both x and y to V_{dd}:
  1. both top P-MOSFETs will be disconnected,
  2. both bottom N-MOSFETS will be connected,
  3. r will be connected to V_{ss}.

Circuit



- Connect both x and y to V_{ss}:
  1. both top P-MOSFETs will be connected,
  2. both bottom N-MOSFETS will be disconnected,
  3. r will be connected to V_{dd}.
- Connect x to V_{dd} and y to V_{ss}:
  1. the upper-most P-MOSFET will be disconnected,
  2. the left-most N-MOSFET will be connected,
  3. r will be connected to V_{ss}.
- Connect x to V_{ss} and y to V_{dd}:
  1. the lower-most P-MOSFET will be disconnected,
  2. the right-most N-MOSFET will be connected,
  3. r will be connected to V_{ss}.
- Connect both x and y to V_{dd}:
  1. both top P-MOSFETs will be disconnected,
  2. both bottom N-MOSFETS will be connected,
  3. r will be connected to V_{ss}.

In summary, we should

1. form a
   - **pull-up network** of P-MOSFET transistors connected to V_{dd},
   - **pull-down network** of N-MOSFET transistors connected to V_{ss},
2. relabel

$$\begin{array}{lcll} V_{ss} & = & 0\text{V} \simeq GND & \models 0 \\ V_{dd} & = & 5\text{V} & \models 1 \end{array}$$

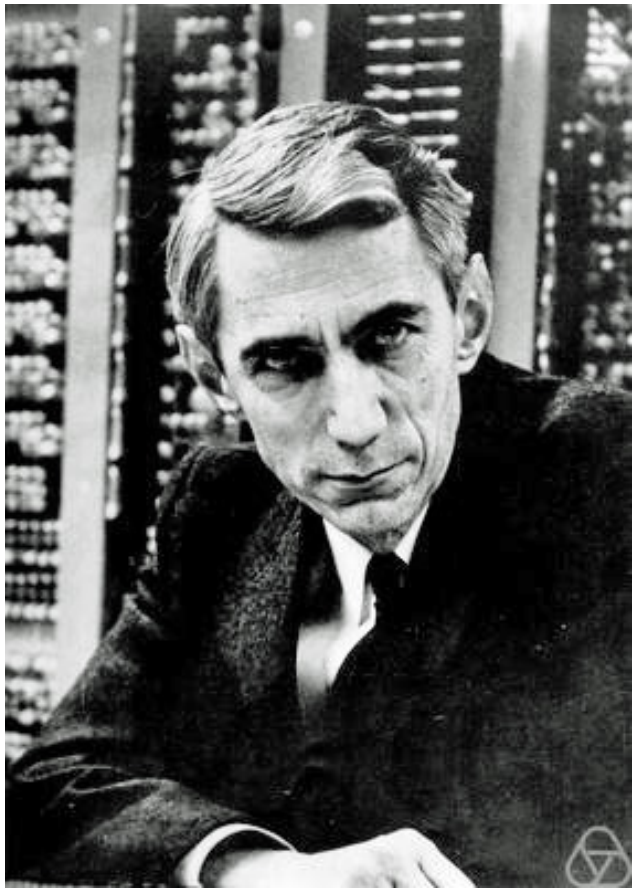
   and assume the power rails are everywhere,
3. specify the functionality of each logic gate using a truth table, e.g.,

x	r
0	1
1	0

x	y	r
0	0	1
0	1	1
1	0	1
1	1	0

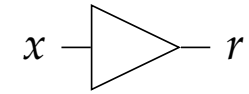
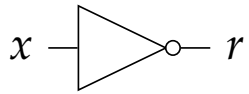
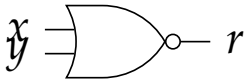
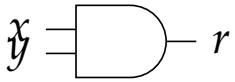
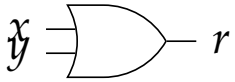
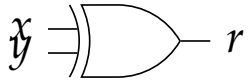
x	y	r
0	0	1
0	1	0
1	0	0
1	1	0

   and represent it using an associated symbol.



https://en.wikipedia.org/wiki/File:ClaudeShannon_MFO3807.jpg

Definition

r is x	$\equiv$	$r = x$	$\equiv$	
r is NOT x	$\equiv$	$r = \neg x$	$\equiv$	
r is x NAND y	$\equiv$	$r = x \overline{\wedge} y$	$\equiv$	
r is x NOR y	$\equiv$	$r = x \overline{\vee} y$	$\equiv$	
r is x AND y	$\equiv$	$r = x \wedge y$	$\equiv$	
r is x OR y	$\equiv$	$r = x \vee y$	$\equiv$	
r is x XOR y	$\equiv$	$r = x \oplus y$	$\equiv$	

Definition

Within some combinatorial logic, two classes of **delay** (which is often described as **propagation delay**, with a hint toward delay of signals more generally) dictate the time between change to some input and corresponding change (if any) in an output: these are

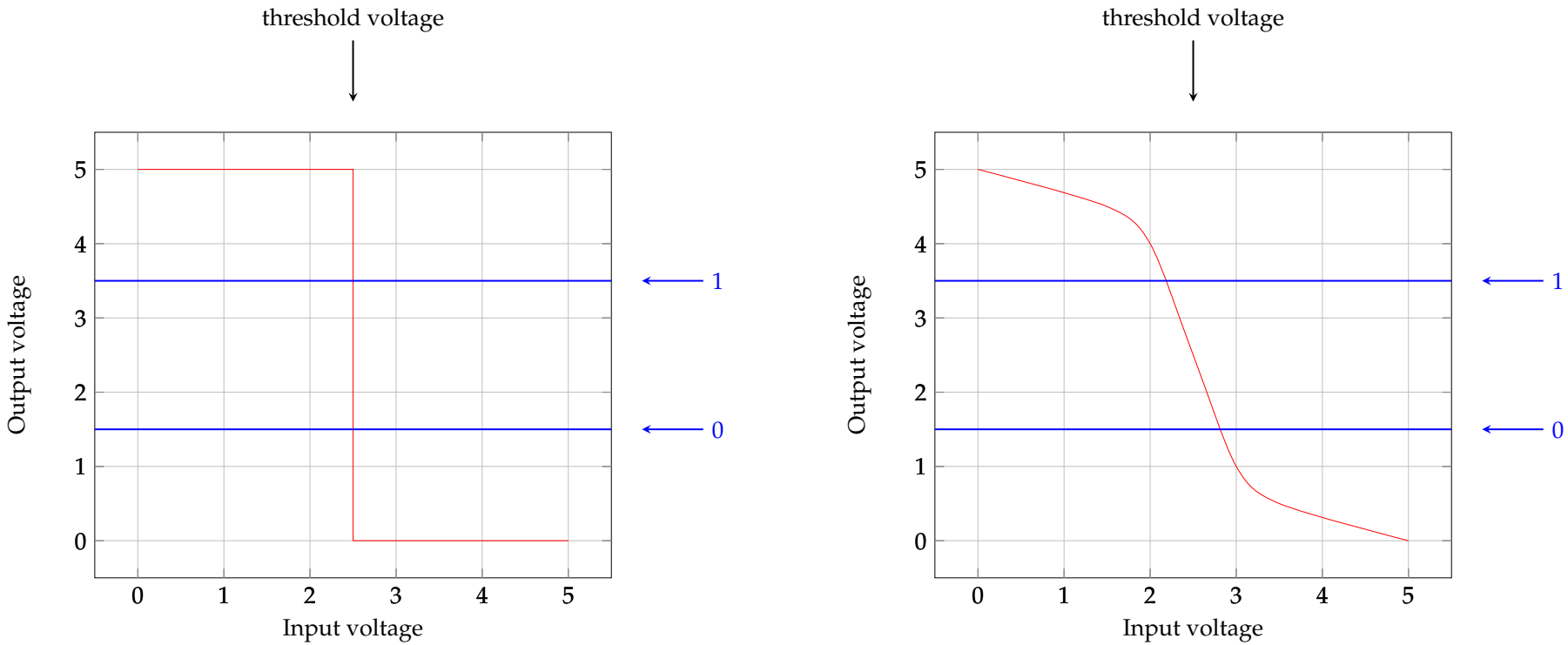
- **wire delay**, which relates to the time taken for current to move through the conductive wire from one point to another, and
- **gate delay**, which relates to the time taken for transistors in each gate to switch between connected and unconnected states.

The latter is typically larger than the former, and both relate to the associated implementations: the latter relates to properties of the transistors used, the former to properties of the wire (e.g., conductivity, length, and so on).

Definition

The **critical path** through some combinatorial logic is the longest sequential path between an input and output, i.e., the path which has the largest total delay (stemming from the individual wire and/or gate delays).

- Example: consider the MOSFET-based NOT gate

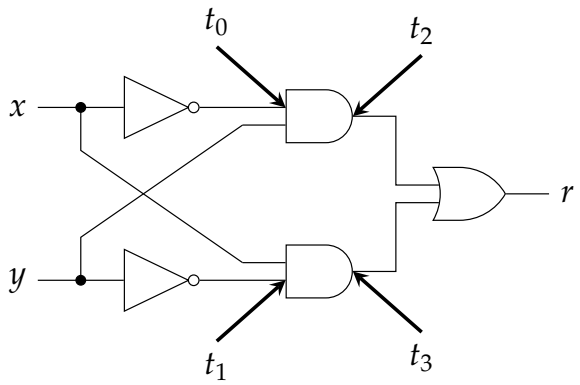


where

- the left-hand side illustrates an idealised, square response, whereas
- the right-hand side illustrates a (more) realistic, curved response.

- Example: *static* (i.e., time-agnostic) evaluation:

Circuit

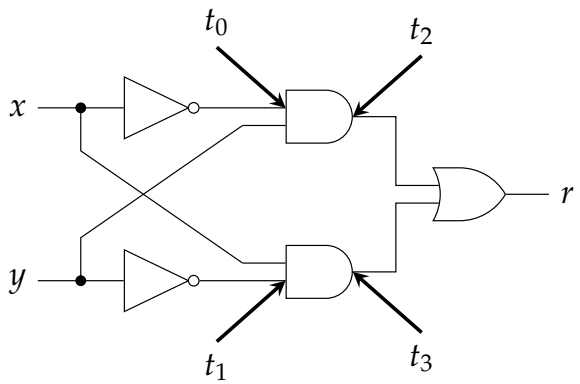


setting $x = 0, y = 1$ means the circuit will compute

$$\begin{array}{lllll} x & = & & = & 0 \\ y & = & & = & 1 \\ t_0 & = & \neg x & = & 1 \\ t_1 & = & \neg y & = & 0 \\ t_2 & = & t_0 \wedge y & = & 1 \\ t_3 & = & t_1 \wedge x & = & 0 \\ r & = & t_2 \vee t_3 & = & 1 \end{array}$$

- Example: *dynamic* (i.e., time-considerate) evaluation:

Circuit

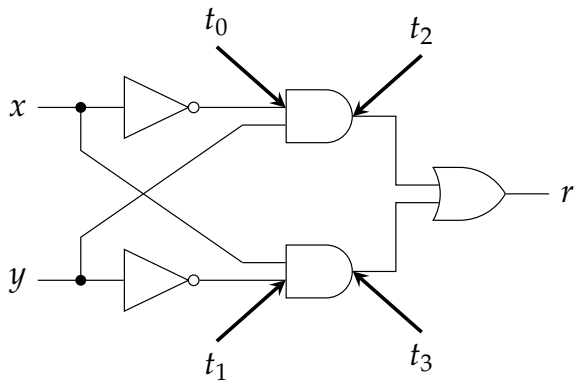


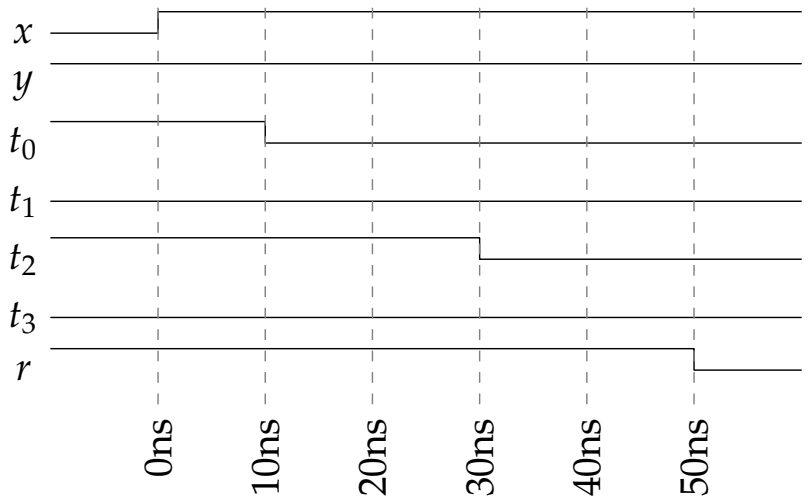
imagine that

$$\begin{aligned} \text{NOT gate} &\rightsquigarrow 10\text{ns} \\ \text{AND gate} &\rightsquigarrow 20\text{ns} \\ \text{OR gate} &\rightsquigarrow 20\text{ns} \end{aligned}$$

- Example: *dynamic* (i.e., time-considerate) evaluation:

Circuit



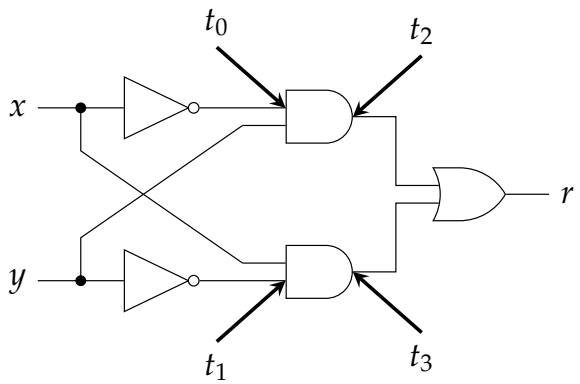


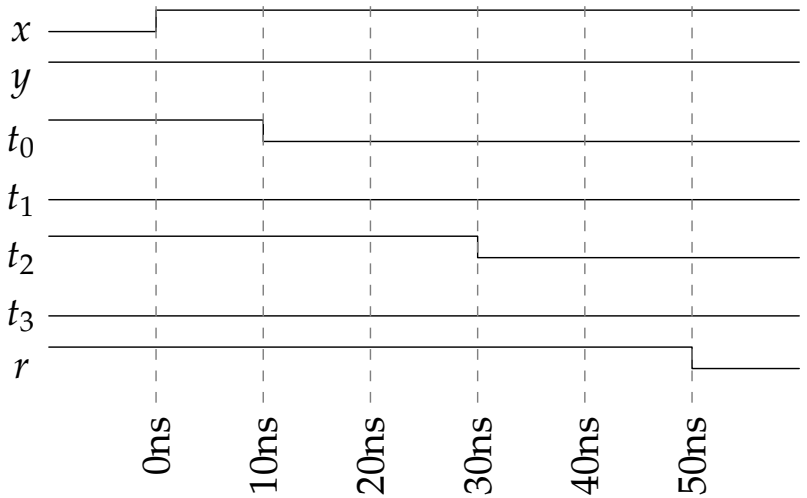
flipping $x = 0, y = 1$ to $x = 1, y = 1$ means the circuit will compute

$$\begin{array}{lcll} x & = & & = 1 \\ y & = & & = 1 \\ t_0 & = & \neg x & = 0 \\ t_1 & = & \neg y & = 0 \\ t_2 & = & t_0 \wedge y & = 0 \\ t_3 & = & t_1 \wedge x & = 0 \\ r & = & t_2 \vee t_3 & = 0 \end{array}$$

- Example: *dynamic* (i.e., time-considerate) evaluation:

Circuit



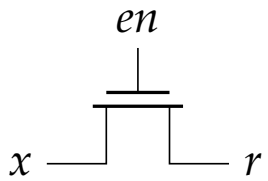


but, crucially,

1. it takes some time matching the **critical path** (i.e, 50ns through a NOT, an AND and an OR gate) to settle into the correct state, so
2. at some points in time, the output doesn't match the inputs.

- Concept: 3-**state logic**, which introduces an "extra" pseudo-value where
  1. 0 represents **false**,
  2. 1 represents **true**, and
  3. *Z* represents **high impedance**,

  can be a useful model of real circuits.
- Think of high impedance as being the null value; the idea is to allow a wire to be "disconnected" per



x	en	r
0	0	**Z**
1	0	**Z**
Z	0	**Z**
0	1	0
1	1	1
Z	1	**Z**
0	**Z**	**Z**
1	**Z**	**Z**
Z	**Z**	**Z**

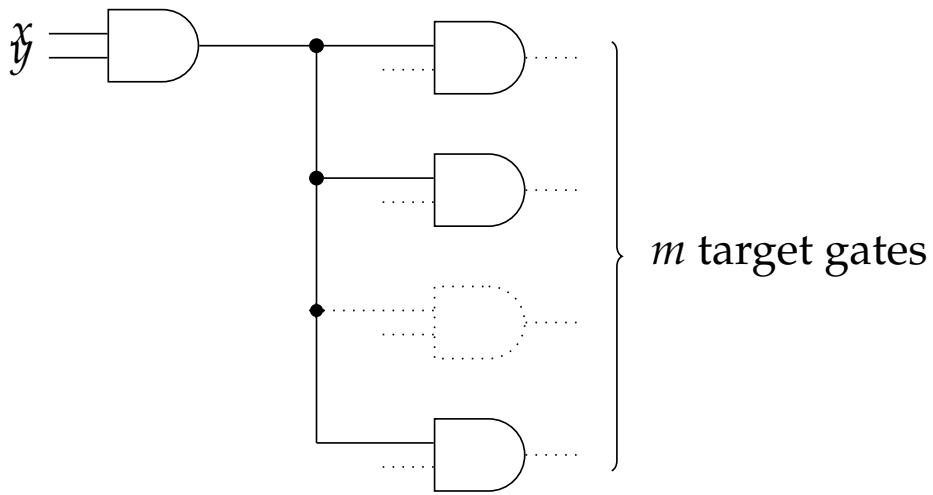
such that the ENABLE gate is really just a switch.

Definition

Consider a given logic gate:

- The term **fan-in** is used to describe the number of inputs to a given gate.
- The term **fan-out** is used to describe the number of inputs (so in a rough sense the number of *other* gates) the output of a given gate is connected to.

Example



Conclusions

- Take away points:
  1. We now have a suite of logic gates, where it's clear

$$\begin{aligned} \textit{behavioural} \text{ properties} &\Leftrightarrow \text{transistors} \\ &\Leftrightarrow \text{semi-conductors} \\ &\Leftrightarrow \text{Physics} \end{aligned}$$

$$\begin{aligned} \textit{functional} \text{ properties} &\Leftrightarrow \text{Boolean algebra} \\ &\Leftrightarrow \text{Mathematics} \end{aligned}$$

  i.e., there's no "magic" steps involved.

Conclusions

- Take away points:

2. Arguably, tougher challenges stem from design and optimisation tasks, i.e., how do we
   - match some specification, e.g., "implement some function f", *and*
   - safisfy some design goals, e.g., "use less than X gates (or Y transistors)" or "ensure a delay less than Zns".
3. Modern transistor design and manufacture imply various constraints: at least a high-level, understanding of these (e.g., Moore's Law etc.) is valuable.

Additional Reading

- *Wikipedia: Transistor*. URL: https://en.wikipedia.org/wiki/Transistor.
- *Wikipedia: Logic gate*. URL: https://en.wikipedia.org/wiki/Logic_gate.
- D. Page. "Chapter 2: Basics of digital logic". In: *A Practical Introduction to Computer Architecture*. 1st ed. Springer, 2009.
- R.J. Smith and R.C. Dorf. "Chapter 12: Transistors and Integrated Circuits". In: *Circuits, Devices and Systems*. 5th ed. Wiley, 1992.
- W. Stallings. "Chapter 11: Digital logic". In: *Computer Organisation and Architecture*. 9th ed. Prentice Hall, 2013.
- A.S. Tanenbaum and T. Austin. "Section 3.1: Gates and Boolean algebra". In: *Structured Computer Organisation*. 6th ed. Prentice Hall, 2012.
- A.S. Tanenbaum and T. Austin. "Section 3.2.1: Integrated circuits". In: *Structured Computer Organisation*. 6th ed. Prentice Hall, 2012.

References

[1] *Wikipedia: Logic gate*. URL: https://en.wikipedia.org/wiki/Logic_gate (see p. 57).

[2] *Wikipedia: Transistor*. URL: https://en.wikipedia.org/wiki/Transistor (see p. 57).

[3] D. Page. "Chapter 2: Basics of digital logic". In: *A Practical Introduction to Computer Architecture*. 1st ed. Springer, 2009 (see p. 57).

[4] R.J. Smith and R.C. Dorf. "Chapter 12: Transistors and Integrated Circuits". In: *Circuits, Devices and Systems*. 5th ed. Wiley, 1992 (see p. 57).

[5] W. Stallings. "Chapter 11: Digital logic". In: *Computer Organisation and Architecture*. 9th ed. Prentice Hall, 2013 (see p. 57).

[6] A.S. Tanenbaum and T. Austin. "Section 3.1: Gates and Boolean algebra". In: *Structured Computer Organisation*. 6th ed. Prentice Hall, 2012 (see p. 57).

[7] A.S. Tanenbaum and T. Austin. "Section 3.2.1: Integrated circuits". In: *Structured Computer Organisation*. 6th ed. Prentice Hall, 2012 (see p. 57).

[8] G.E. Moore. "Cramming more components onto integrated circuits". In: *Electronics Magazine* 38.8 (1965), pp. 114–117 (see pp. 25–27).