ELEC3221

Digital IC & Sytems Design

Iain McNally

≈ 15 lectures

Koushik Maharatna

≈ 15 lectures

Digital IC & Sytems Design

- Assessment

10%	Coursework	L-Edit Gate Design (BIM)
90%	Examination	

- Books

Integrated Circuit Design

a.k.a. Principles of CMOS VLSI Design - A Circuits and Systems Perspective

Neil Weste & David Harris
Pearson, 2011

Digital System Design with SystemVerilog

Mark Zwolinski
Pearson Prentice-Hall, 2010

Digital IC & Sytems Design

Iain McNally

Integrated Circuit Design

- Content
  - Introduction
  - Overview of Technologies
  - Layout
  - CMOS Processing
  - Design Rules and Abstraction
  - Cell Design and Euler Paths
  - System Design using Standard Cells
  - Pass Transistor Circuits
  - Latches and Flip-Flops
  - PLAs, ROMs, RAMs
  - Wider View
- Notes & Resources

  `https://secure.ecs.soton.ac.uk/notes/bim/notes/icd/`

History

1947 First Transistor

John Bardeen, Walter Brattain, and William Shockley (Bell Labs)

1952 Integrated Circuits Proposed

Geoffrey Dummer (Royal Radar Establishment) - *prototype failed...*

1958 First Integrated Circuit

Jack Kilby (Texas Instruments) - *Co-inventor*

1959 First Planar Integrated Circuit

Robert Noyce (Fairchild) - *Co-inventor*

1961 First Commercial ICs

Simple logic functions from TI and Fairchild

1965 Moore’s Law

Gordon Moore (Fairchild) observes the trends in integration.

Moore's Law

Predicts exponential growth in the number of components per chip.

1965 - 1975 Doubling Every Year

In 1965 Gordon Moore observed that the number of components per chip had doubled every year since 1959 and predicted that the trend would continue through to 1975.

Moore describes his initial growth predictions as "ridiculously precise".

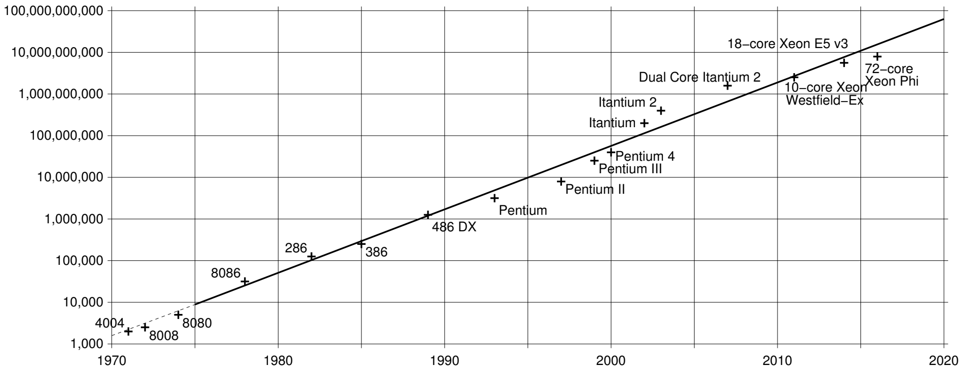
1975 - 201? Doubling Every Two Years

In 1975 Moore revised growth predictions to doubling every two years.

Growth would now depend only on process improvements rather than on more efficient packing of components.

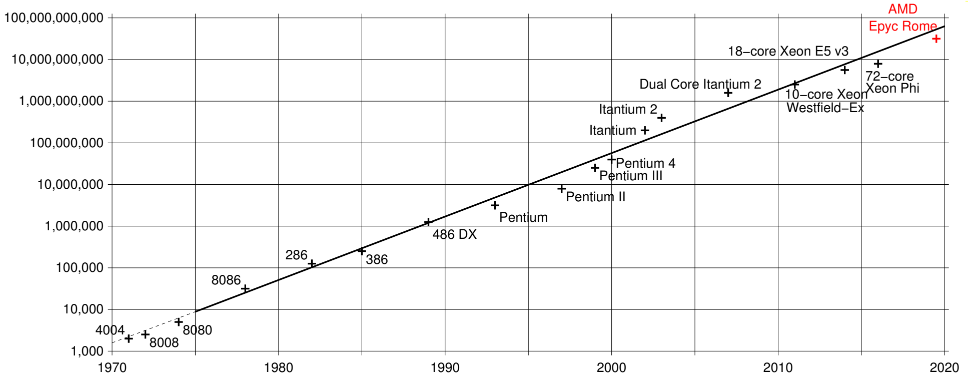
In 2000 he predicted that the growth would continue at the same rate for another 10-15 years before slowing due to physical limits.

Moore's Law at Intel[1]



[1] Intel was founded by Gordon Moore and Robert Noyce from Fairchild

Moore's Law at Intel[2] + AMD/TSMC



[2] Intel has had some hiccups recently - AMD/TSMC appear to be in front

Moore's Law; a Self-fulfilling Prophesy

The whole industry has used the Moore's Law curve to plan new fabrication facilities.

Slower - wasted investment

Must keep up with the Joneses[3].

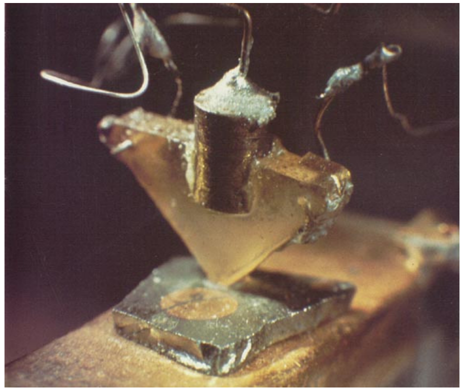
Faster - too costly

Cost of capital equipment to build ICs doubles approximately every 4 years.

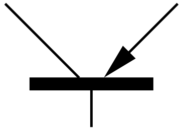
Moore's law is not dead (at least not quite). As transistor dimensions approach the size of a few tens of molecules, new techniques are needed. Recent developments include the stacking of transistors in V-NAND Flash memory to achieve higher densities.

[3] or the Intels

1947 Point Contact transistor

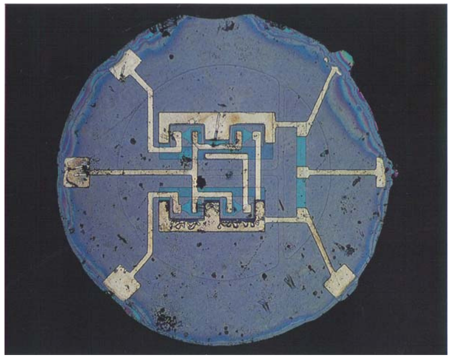


Source: Bell Labs

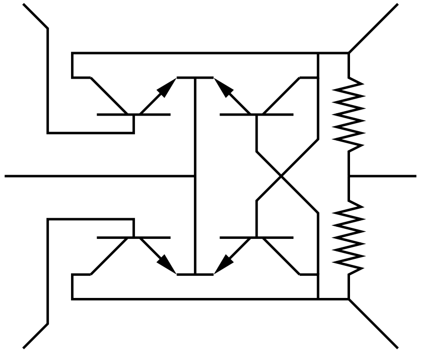


1961 Fairchild Bipolar RTL RS Flip−Flop

(4 Transistors)

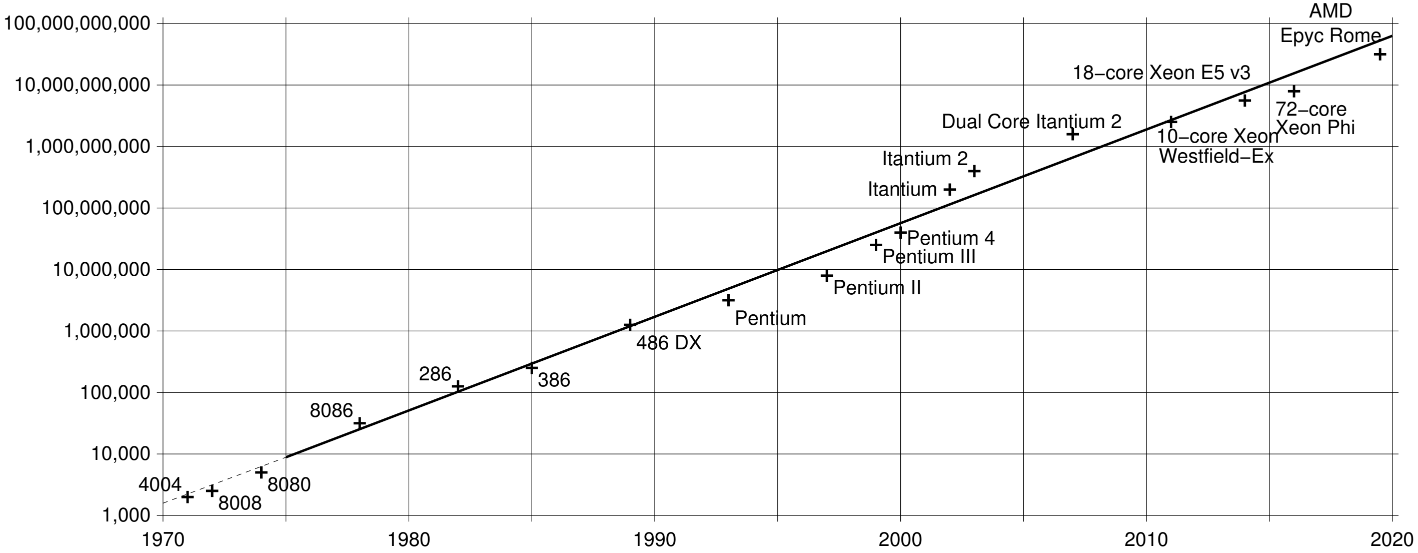


Source: Fairchild



Moore’s Law (1965) Number of components has doubled every year and will continue to do so until 1975

Moore’s Law (1975) Number of components will double every two years

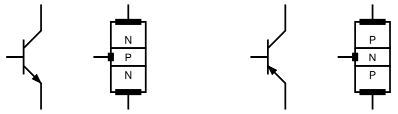


Self−fulfilling Prophecy

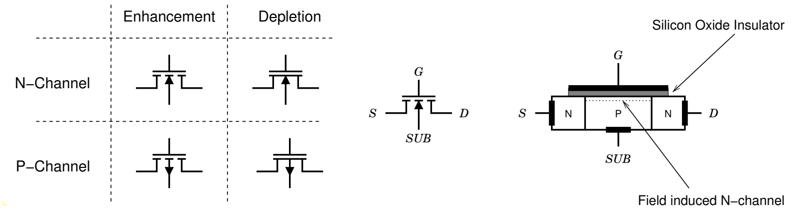
Components for Logic

Diode

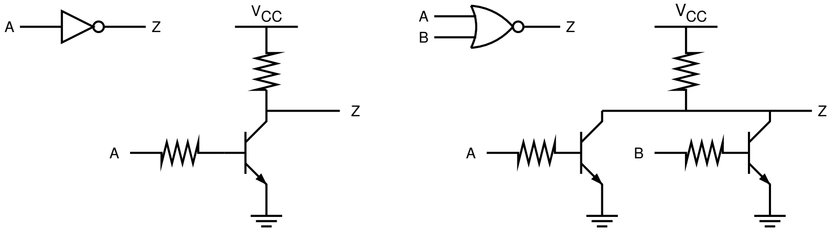
Bipolar Transistors



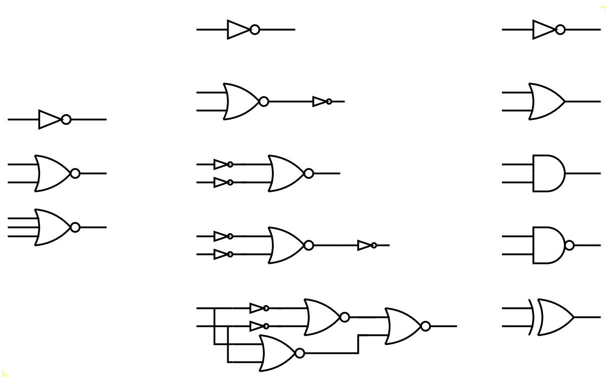
MOS Transistors



RTL Inverter and NOR gate



All functions can be realized using only the NOR gates[1] available in the RTL logic family.[2]

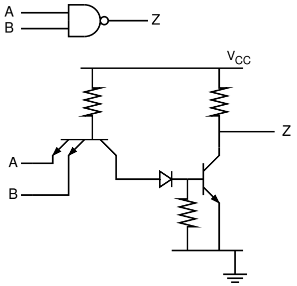


[1] Note that an inverter is a special case of a NOR gate with only one input.

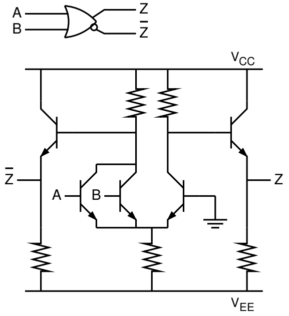
[2] NAND gates could be used instead for logic families which support only NAND gates.

Other Bipolar Technologies

TTL NAND Gate



ECL OR/NOR Gate

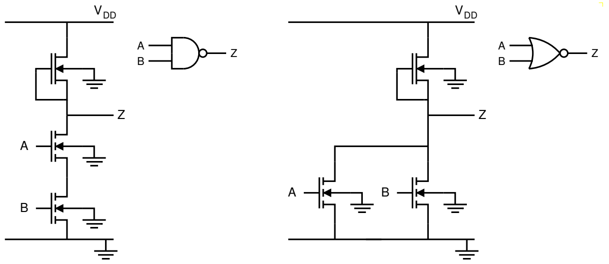


- TTL gives faster switching than RTL at the expense of greater complexity[3]. The characteristic multi-emitter transistor reduces the overall component count.
- ECL is a very high speed, high power, non-saturating technology.

[3] Most TTL families are more complex than the basic version shown here

Overview of Technologies

NMOS - a VLSI technology.

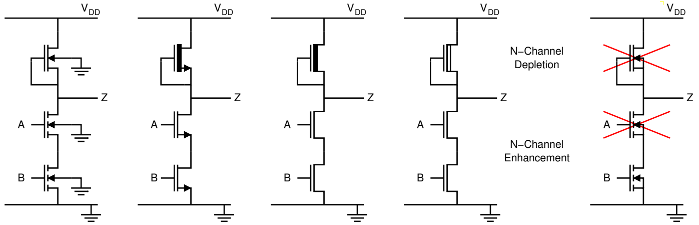


- Circuit function determined by series/parallel combination of devices.
- Depletion transistor acts as non-linear load resistor.

  Resistance increases as the enhancement device turns on, thus reducing power consumption.
- The low output voltage is determined by the size ratio of the devices.

Overview of Technologies

Alternative transistors representations for NMOS circuits



Various shorthands are used for simplifying NMOS circuit diagrams.

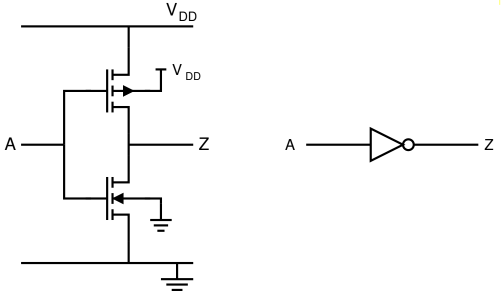
- Substrate connections need not be drawn since all must connect to Gnd.
- Since source and drain are indistinguishable in the layout, there is no need to show the source on the circuit diagram.

Note that schematic tools not designed for IC design will usually include inappropriate 3-terminal symbols.

Overview of Technologies

CMOS logic

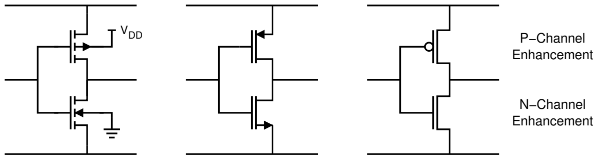
CMOS - *state of the art* VLSI.



- An active PMOS device complements the NMOS device giving:
  - rail to rail output swing.
  - negligible static power consumption.

Digital CMOS Circuits

Alternative transistor representations for CMOS circuits



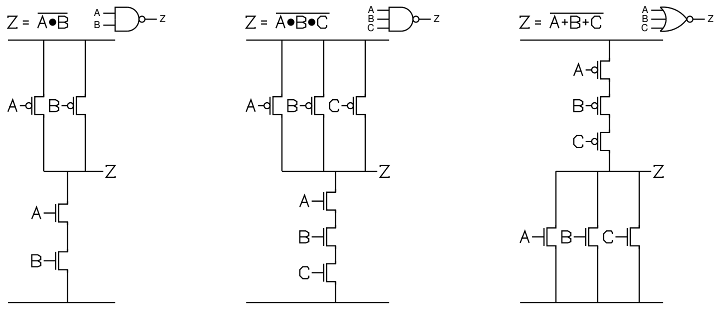
Digital CMOS circuits[4] tend to use simplified symbols like their NMOS counterparts.

- In general substrate connections are not drawn where they connect to Vdd (PMOS) and Gnd (NMOS).
- All CMOS devices are enhancement mode.
- Transistors act as simple digitally controlled switches.

[4] in analog CMOS circuits we may have wells not connected to Vdd/GND

Digital CMOS Circuits

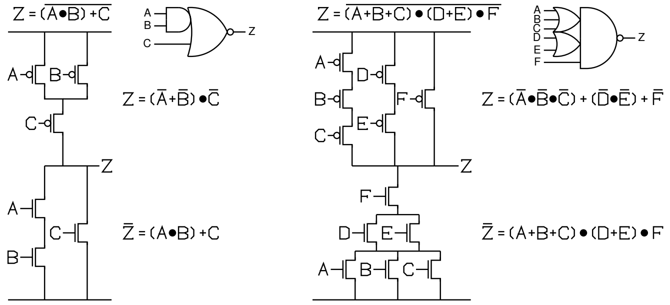
Static CMOS complementary gates


$Z = \overline{A \bullet B}$
A
B
Z
$Z = \overline{A \bullet B \bullet C}$
A
B
C
Z
$Z = \overline{A + B + C}$
A
B
C
Z

- For any set of inputs there will exist either a path to Vdd or a path to Gnd.

Digital CMOS Circuits

Compound Gates


$Z = \overline{(A \bullet B) + C}$
A
B
C
Z
$Z = \overline{(A+B+C) \bullet (D+E) \bullet F}$
A
B
C
D
E
F
Z

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

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

$Z = (\bar{A} \bullet \bar{B} \bullet \bar{C}) + (\bar{D} \bullet \bar{E}) + \bar{F}$

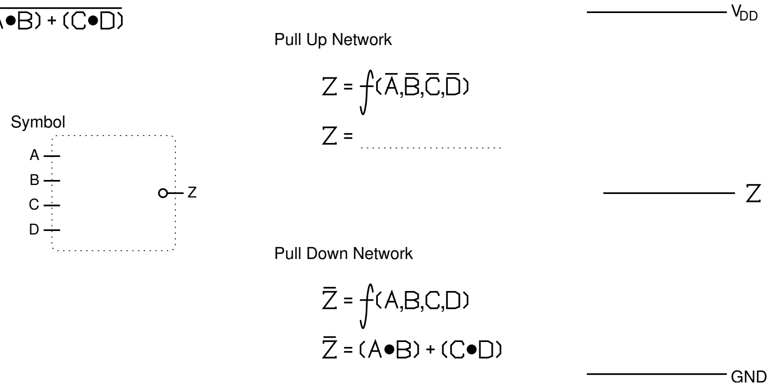
$\bar{Z} = (A+B+C) \bullet (D+E) \bullet F$

- All compound gates are inverting.
- Realisable functions are arbitrary AND/OR expressions with inverted output.

Digital CMOS Circuits

Compound Gate Example

$Z = \overline{(A \bullet B) + (C \bullet D)}$



Pull Up Network

$Z = f(\bar{A}, \bar{B}, \bar{C}, \bar{D})$

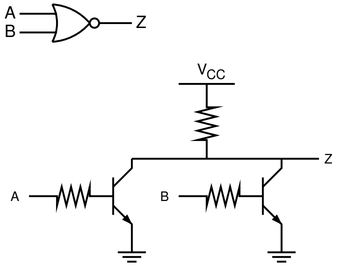
Z = ...

Pull Down Network

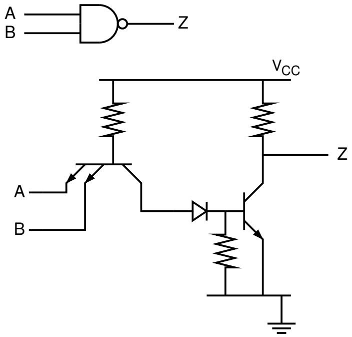
$\bar{Z} = f(A, B, C, D)$

$\bar{Z} = (A \bullet B) + (C \bullet D)$

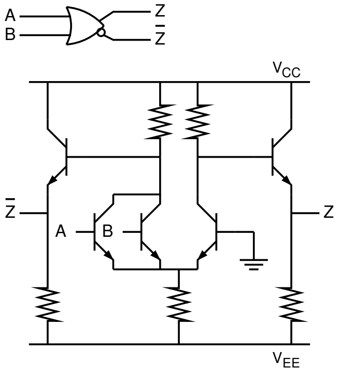
RTL NOR Gate



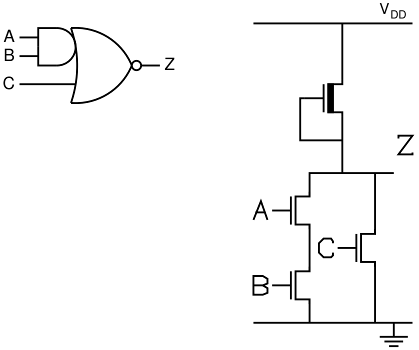
TTL NAND Gate



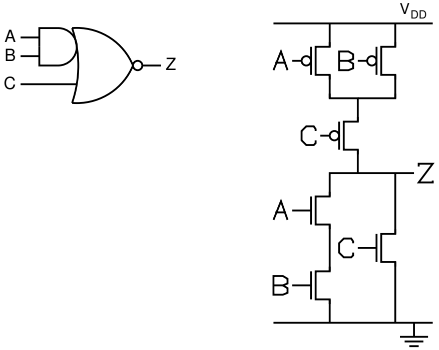
ECL OR/NOR Gate



NMOS Compund Gate



CMOS Compund Gate



- Bipolar Transitors with Resistors - MSI/LSI

RTL - NOR TTL - NAND ECL - OR/NOR

- MOS Transistors (no resistors) - VLSI

NMOS CMOS - No static power!

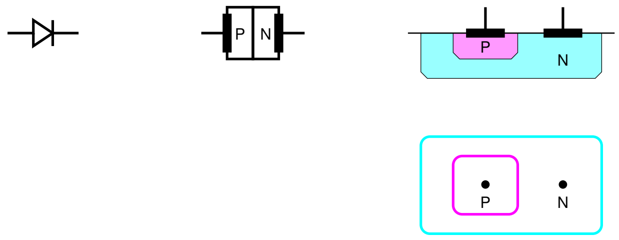
Both allow construction of NOR, NAND & Compound gate (always inverting)

2000

Components for IC Design

Diodes and Bipolar Transistors

Diode

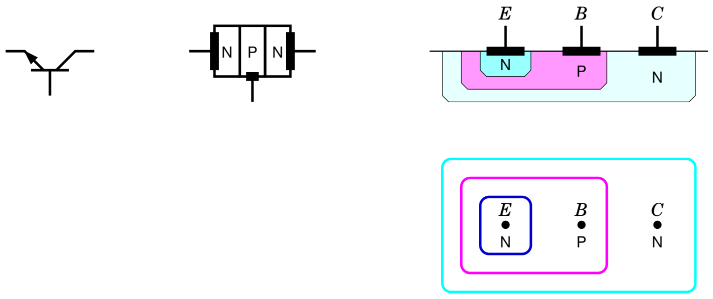


- Ideal structure - 1D
- Real structure - 3D
- Depth controlled implants.

Components for IC Design

Diodes and Bipolar Transistors

NPN Transistor

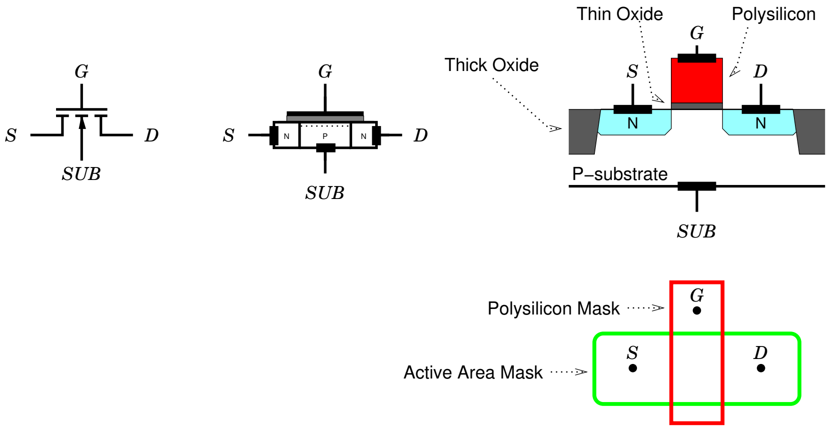


- Two n-type implants.

Components for IC Design

MOS Transistors

Simple NMOS Transistor

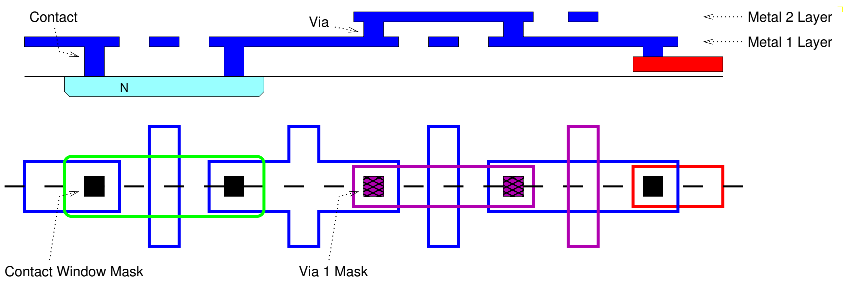


Components for IC Design

Simple NMOS Transistor

- Active Area mask defines extent of *Thick Oxide*.
- Polysilicon mask also controls extent of *Thin Oxide* (alias *Gate Oxide*).
- N-type implant has no extra mask.
  - It is blocked by thick oxide and by polysilicon.
  - The implant is *Self Aligned*.
- Substrate connection is to bottom of wafer.
  - All substrates to ground.
- Gate connection not above transistor area.
  - Design Rule.

Interconnect



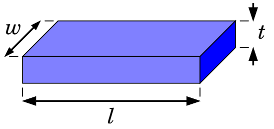
Conductors: Polycrystaline Silicon (Polysilicon/Poly) 🟥
Diffusion (controlled by Active Area Mask) 🟩
Metal 1 🟦
Metal 2 🟪

- Crossing conductors on different masks do not interact[1].
  - Explicit contact/via is required for connection.
- Crossing conductors on the same mask are always connected.

[1] the exception to this rule is that polysilicon crossing diffusion gives us a transistor

Interconnect

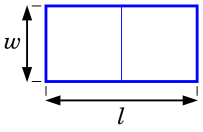
Resistance



$$R = \left(\frac{\rho}{t}\right)\left(\frac{l}{w}\right)$$

where ρ is the resistivity constant
$3.2 \times 10^{-8}\Omega m$ for aluminium
$1.7 \times 10^{-8}\Omega m$ for copper

Since t and ρ are fixed for a paricular mask layer, the value that is normally used is the sheet resistance: $R_s = \left(\frac{\rho}{t}\right)$.



$$R = R_s\left(\frac{l}{w}\right)$$

where R_s is sheet resistance
$0.1\Omega/\square$ for $170nm$ thick copper

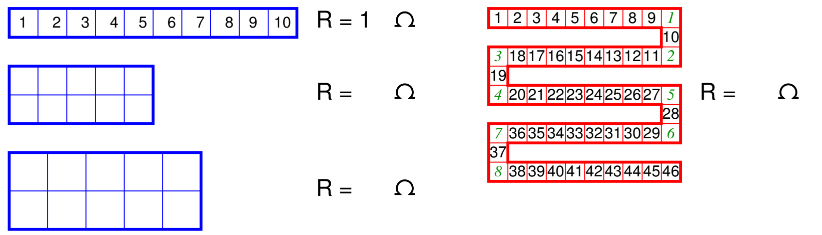
R_s = resistance of a square (i.e. $w = l$) so the units for R_s are $\Omega/\square$ (ohms per square).

Components for IC Design

Resistors

Examples for Metal
assuming Rs = 0.1 ohms per square

Example for Polysilicon
assuming Rs = 200 ohms per square

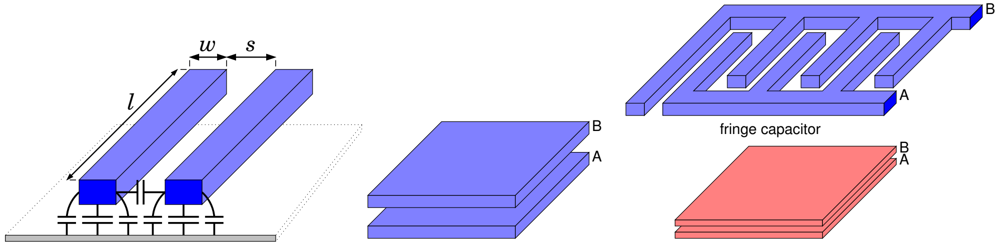


- for larger resistances we need minimum width poly (often combined with a *serpantine* shape) to save on area
- corner squares count as half[2] squares
- for predicatability and matching we may need wider tracks without corners

[2] effective resistance ≈ $0.56R_s$

Components for IC Design

Capacitors



parallel tracks over underlying conductor
(e.g. bulk silicon)

metal–insulator–metal capacitor

fringe capacitor

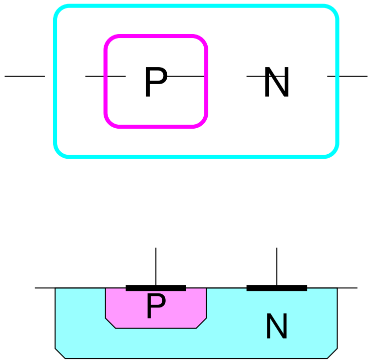
poly–insulator–poly capacitor
(specialist analog process with 2 poly layers)

- Capacitance to underlying conductor $C = C_a\, w\, l + 2\, C_f\, l$
- Coupling capacitance to adjacent track $C = C_c\, l/s$

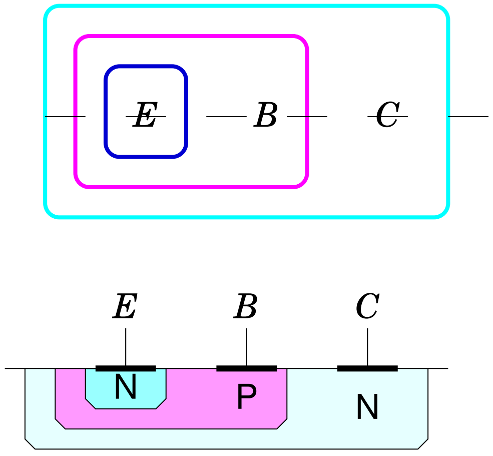
  where C_a, C_f, C_c are constants for a given layer and process

  *in digital designs our only aim is to minimise **parasitic** capacitance*

Diode

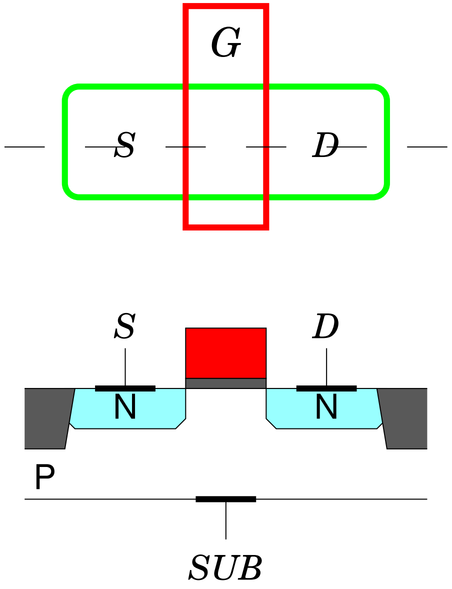


NPN Transistor

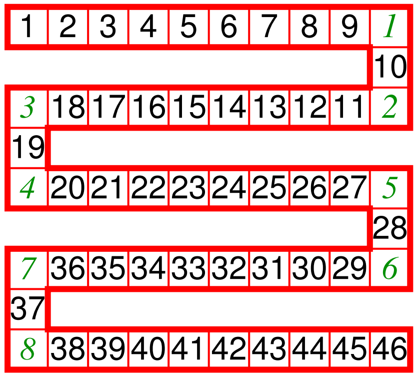


NMOS Enhancement transistor

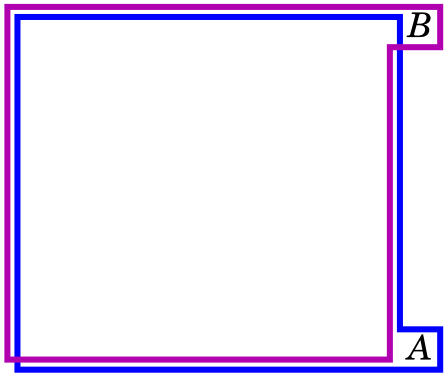
NMOS Process

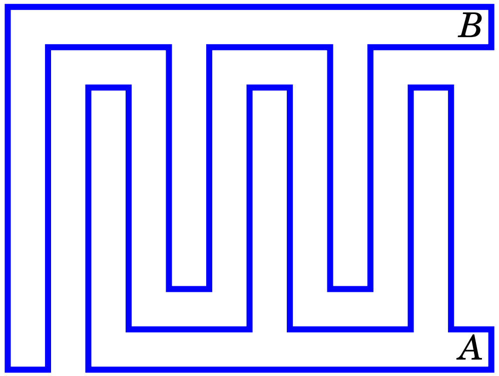


Resistor



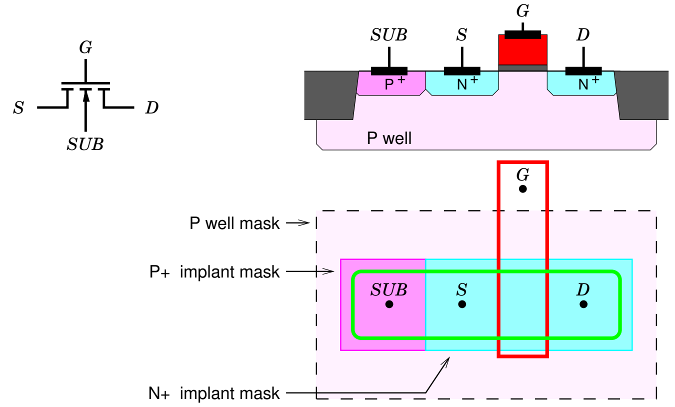
Capacitors





3000

NMOS Transistor – with top substrate connection

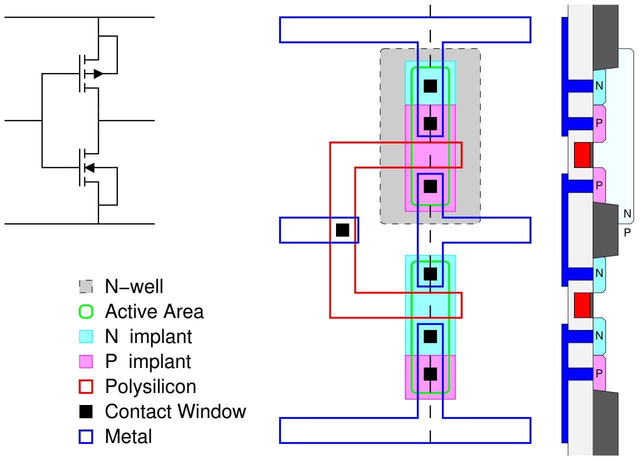


NMOS Transistor – with top substrate connection

Where it is not suitable for substrate connections to be shared, a more complex process is used.

- Five masks must be used to define the transistor:
  - P Well
  - Active Area
  - Polysilicon
  - N+ implant
  - P+ implant
- P Well, for isolation.
- Top *substrate* connection.
- P+/N+ implants produce good *ohmic* contacts.

CMOS Inverter



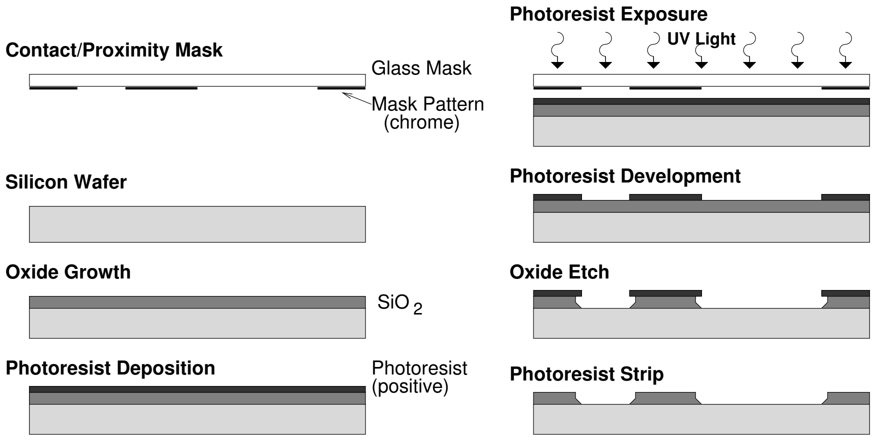
CMOS Inverter

- The process described here is an *N Well process* since it has only an N Well.

  P Well and Twin Tub processes also exist.
- Note that the P-N junction between chip substrate and N Well will remain reverse biased.

  Thus the transistors remain isolated.
- N implant defines NMOS source/drain and PMOS substrate contact.
- P implant defines PMOS source/drain and NMOS substrate contact.

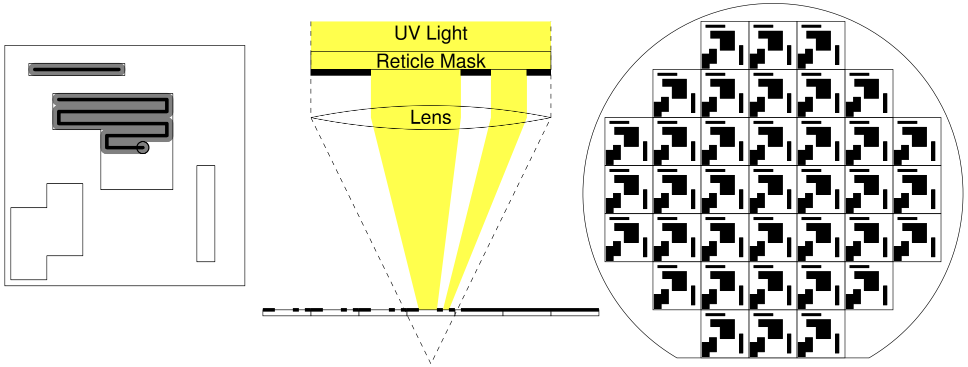
Processing – Photolithography



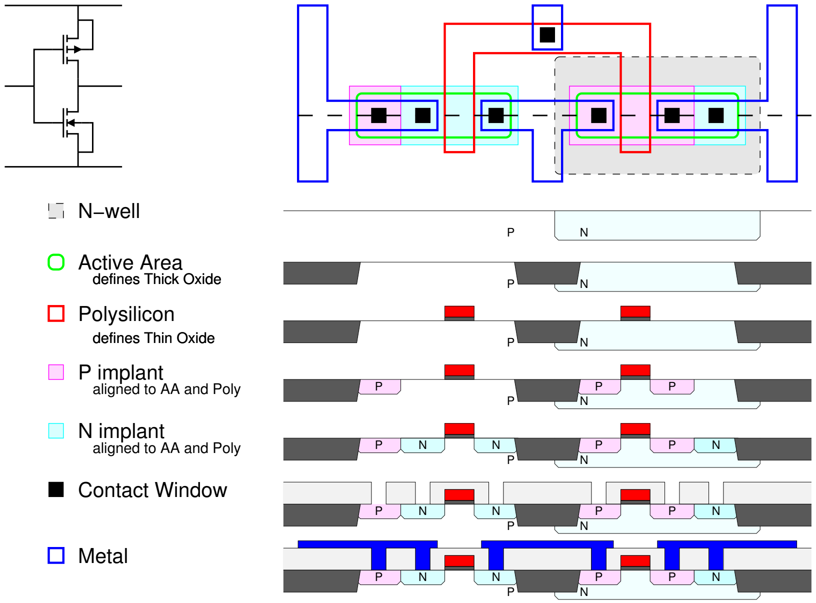
Processing – Mask Making

Reticle written by scanning electron beam

Pattern reproduced on wafer (or contact/proximity mask) by step and repeat with optical reduction



- Optical reduction allows narrower line widths.



▨ N–well

🟩 Active Area
defines Thick Oxide

🟥 Polysilicon
defines Thin Oxide

▨ P implant
aligned to AA and Poly

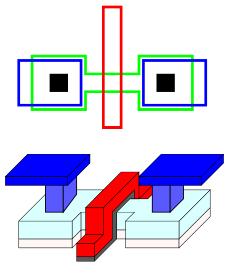
▨ N implant
aligned to AA and Poly

■ Contact Window

🟦 Metal

CMOS - Short Gate Techiques

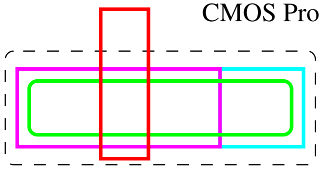
Fin FET



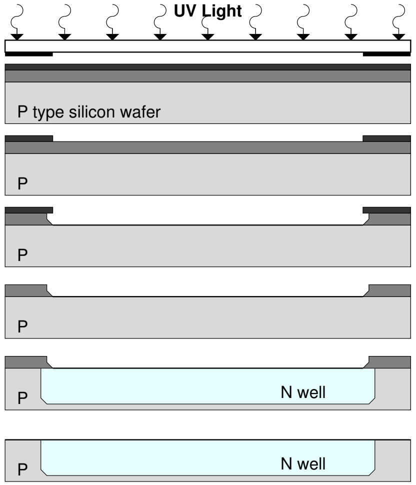
- With the aid of trenches we raise the active area above the bulk silicon.
- We can then wrap the gate around the channel.
- Avoids an effect where a channel is created in a region which is closer to the drain than the gate.

PMOS Enhancement transistor

CMOS Process

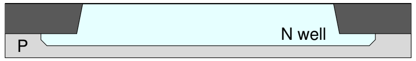


N Well

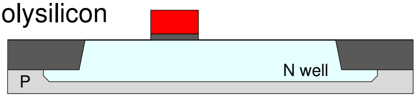


many steps for a single mask

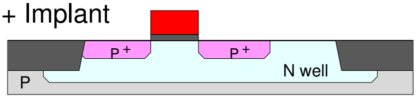
Active Area



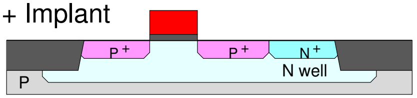
Polysilicon



P+ Implant

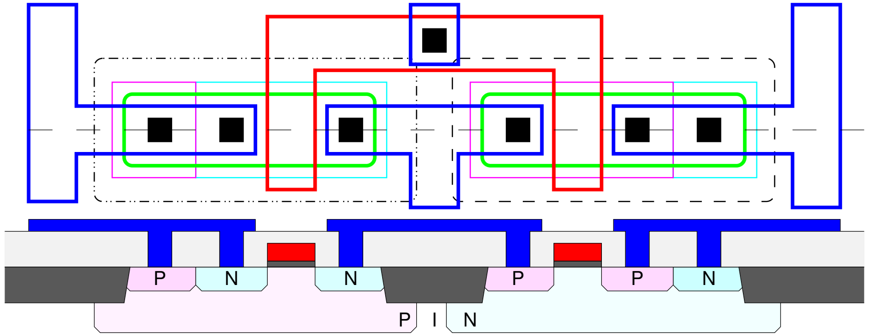


N+ Implant



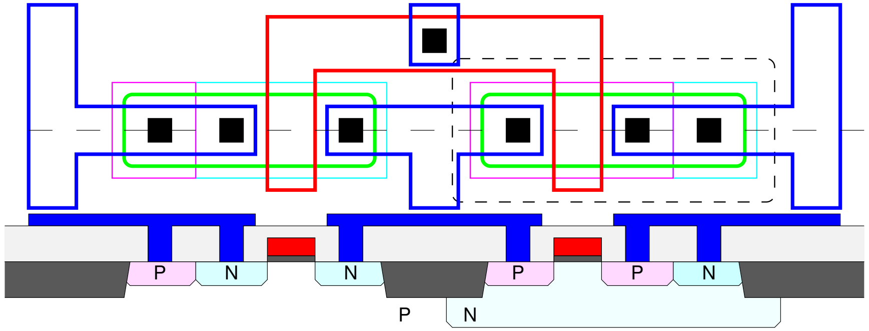
CMOS Inverter

Twin Tub CMOS Process



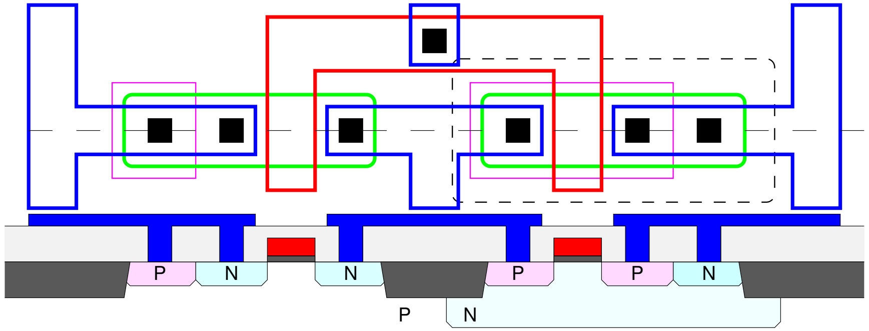
CMOS Inverter

N−Well CMOS Processs (with explicit N+ implant mask)



CMOS Inverter

N−Well CMOS Processs (without explicit N+ implant mask)



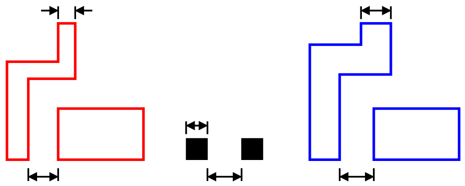
Features may be determined by a number of masks

e.g. NMOS source drain: ActiveArea AND NOT(NWell OR Poly OR PImplant)

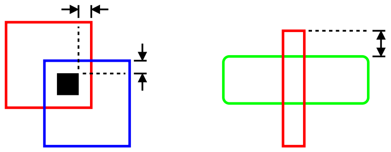
Design Rules

To prevent chip failure, designs must conform to design rules:

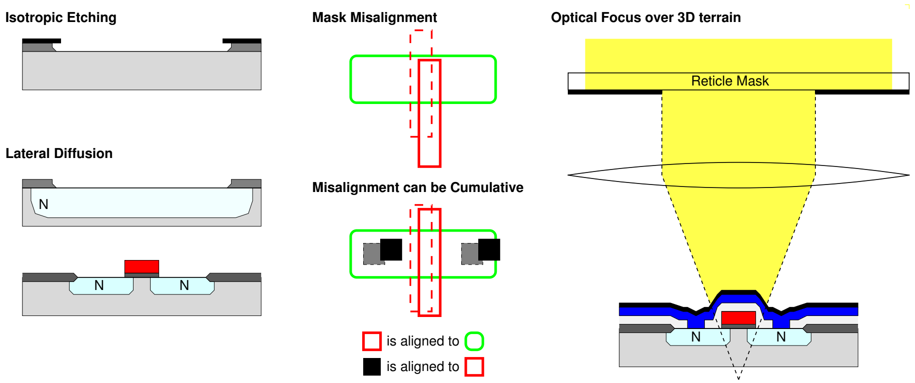
- Single layer rules



- Multi-layer rules

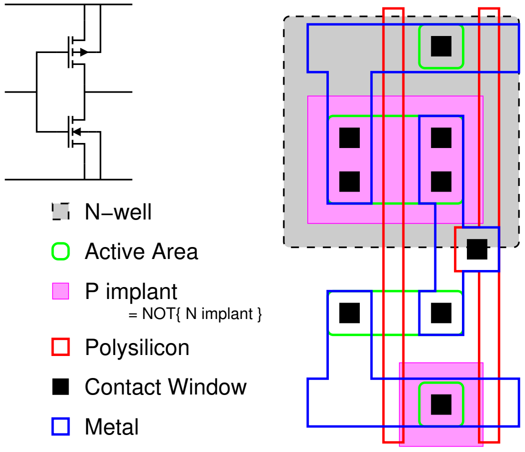


Derivation of Design Rules



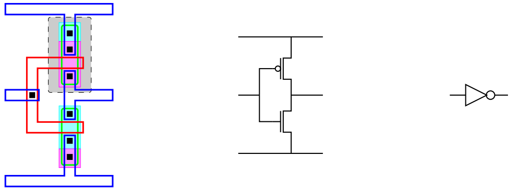
Design Rules

0.5 μm CMOS inverter



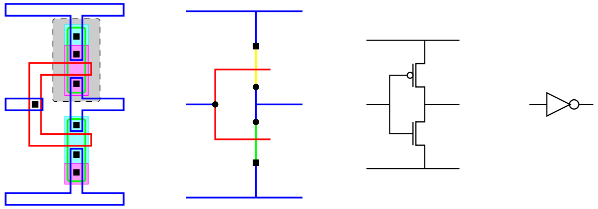
Abstraction

Levels of Abstraction



- Mask Level Design
  - Laborious Technology/Process dependent.
  - Design rules may change during a design!
- Transistor Level Design
  - Process independent, Technology dependent.
- Gate Level Design
  - Process/Technology independent.

Abstraction - Stick Diagrams



Stick diagrams give us many of the benefits of abstraction:

- Much easier/faster than full mask specification.
- Process independent (valid for any CMOS process).
- Easy to change.

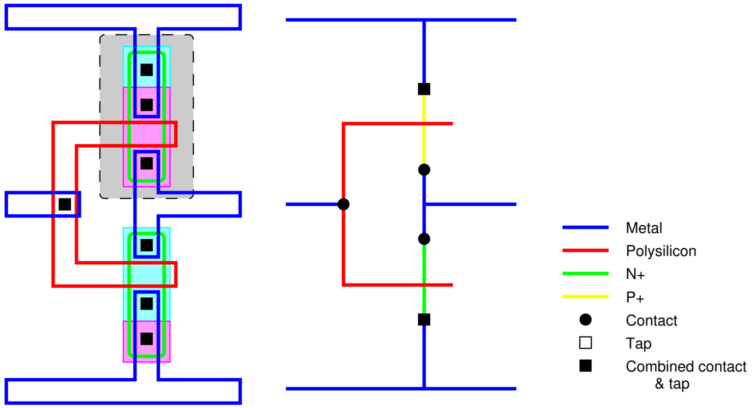
while avoiding some of the problems:

- Optimized layout may be generated much more easily from a stick diagram than from transistor or gate level designs.[1]

[1] note that all IC designs must end at the mask level.

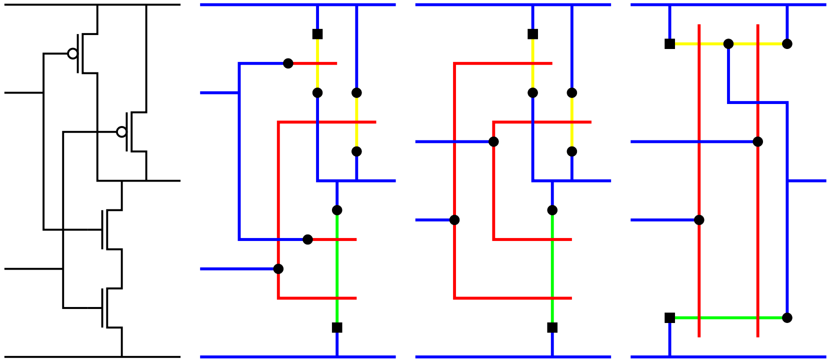
Digital CMOS Design

Stick Diagrams



Digital CMOS Design

Stick Diagrams

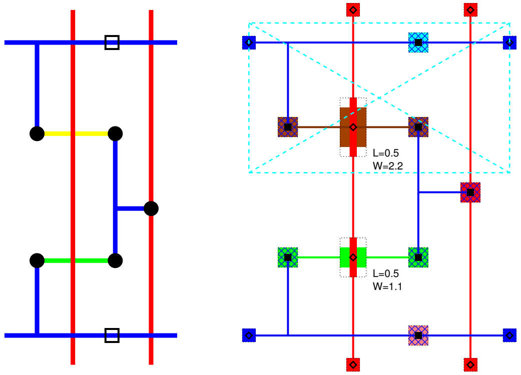


Digital CMOS Design

Stick Diagrams

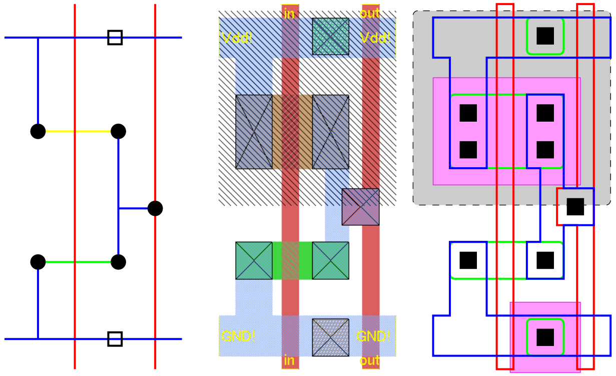
- *Explore your Design Space.*
  - Implications of crossovers.
  - Number of contacts.
  - Arrangement of devices and connections.
- Process independent layout.
- Easy to expand to a full layout for a particular process.

Sticks and CAD - Symbolic Capture



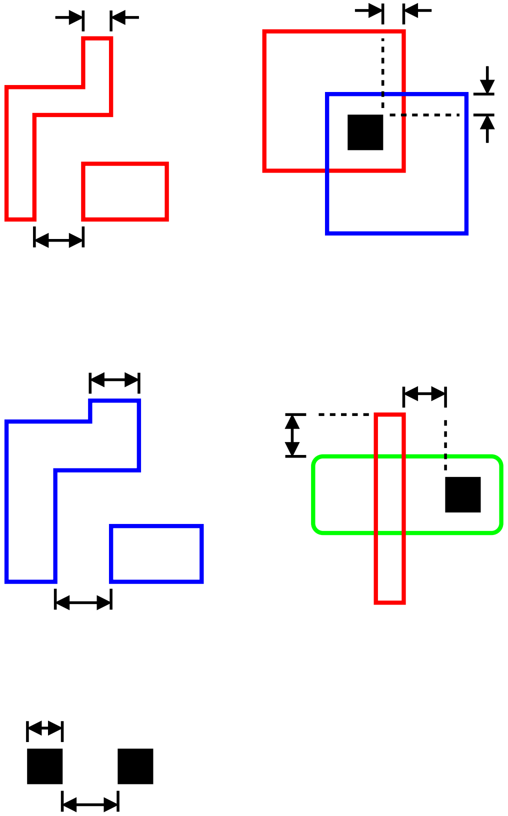
- Transistors are placed and explicitly sized.
  - components are joined with zero width wires.
  - contacts are automatically selected as required.
- A semi-automatic compaction process will create DRC correct layout.

Sticks and CAD - Magic

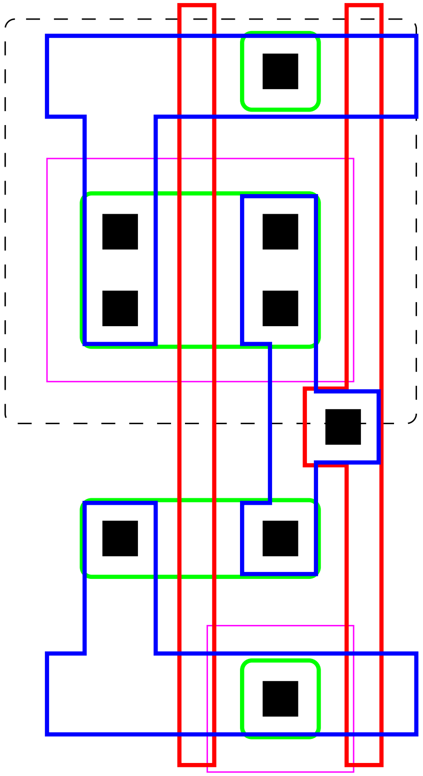


- Log style design (sticks with width) - DRC errors are flagged immediately.
  - again contacts are automatically selected as required.
- On-line DRC leads to rapid generation of correct designs.
  - symbolic capture style compaction is available if desired.

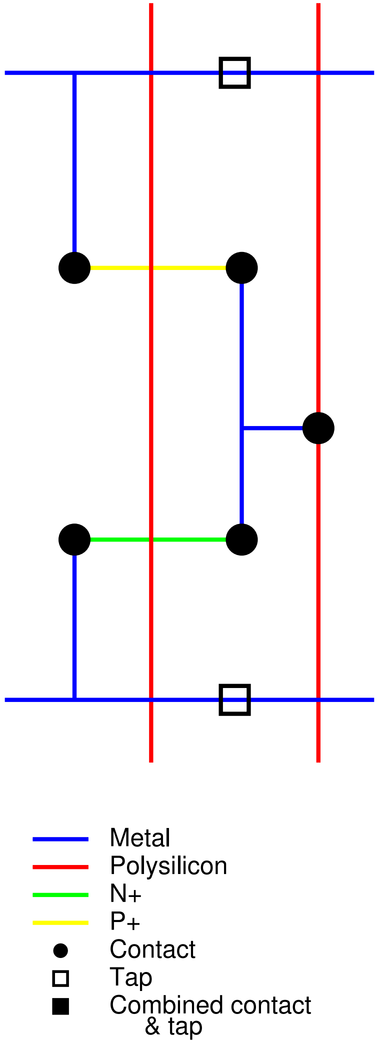
Design Rules – width, separation, overlap



Optimised Mask Layout



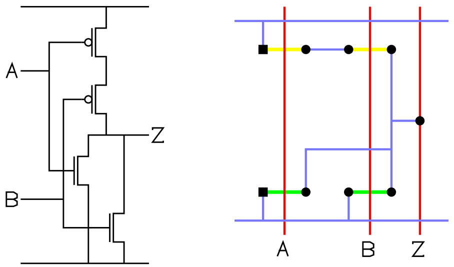
Equivalent Stick Diagram



5000

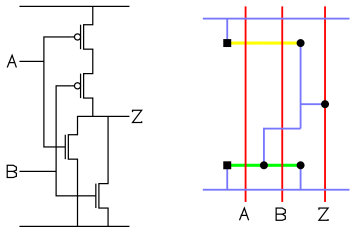
A logical approach to gate layout.

- *All complementary gates may be designed using a single row of n-transistors above or below a single row of p-transistors, aligned at common gate connections.*



Euler Path

- For the majority of these gates we can find an arrangement of transistors such that we can butt adjoining transistors.
  - Careful selection of transistor ordering.
  - Careful orientation of transistor source and drain.
- Referred to as *line of diffusion*.



Finding an Euler Path

Computer Algorithms

- It is relatively easy for a computer to consider all possible arrangements of transistors in search of a suitable Euler path.

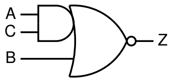
  This is not so easy for the human designer.

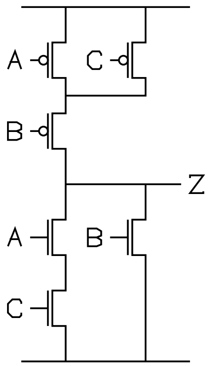
One Human Algorithm

- Find a path which passes through all n-transistors exactly once.
- Express the path in terms of the gate connections.
- Is it possible to follow a similarly labelled path through the p-transistors?
  - Yes – you've succeeded.
  - No – try again (you may like to try a p path first this time).

Finding an Euler Path

$Z = \overline{(A \bullet C) + B}$



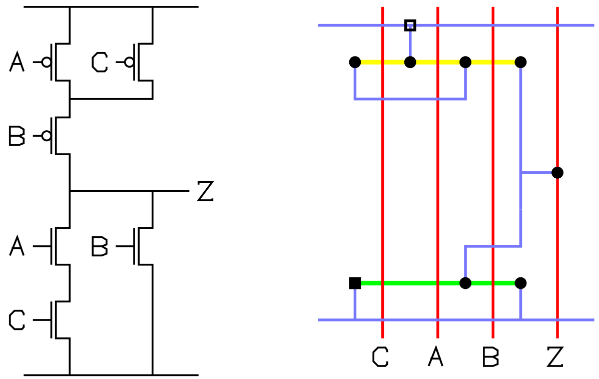


A−C−B
B−A−C
B−C−A
C−A−B ⇐

A−B−C
A−C−B
B−A−C
B−C−A
C−A−B ⇐
C−B−A

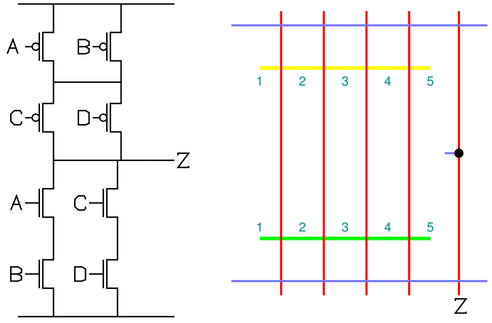
Here there are four possible Euler paths.

Finding an Euler Path



Euler Path Example

$Z = \overline{(A \bullet B) + (C \bullet D)}$



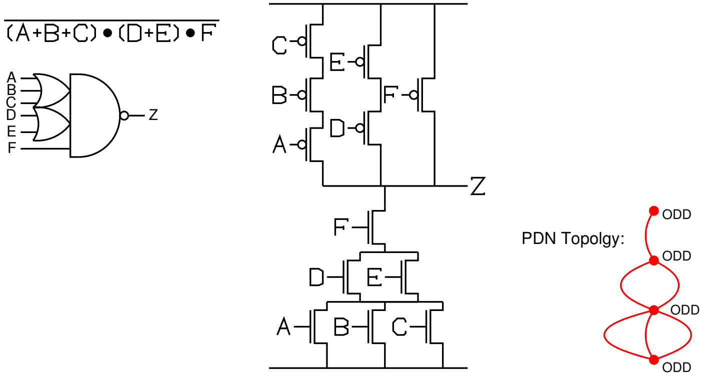
1. Find Euler path
2. Label poly columns
3. Route power nodes
4. Route output node
5. Route remaining nodes
6. Add taps[1] for PMOS and NMOS

A combined contact and tap, ■, may be used only where a power contact exists at the end of a line of diffusion. Where this is not the case a simple tap, □, should be used.

[1] 1 tap is good for about 6 transistors – insufficient taps may leave a chip vulnerable to latch-up

Finding an Euler Path

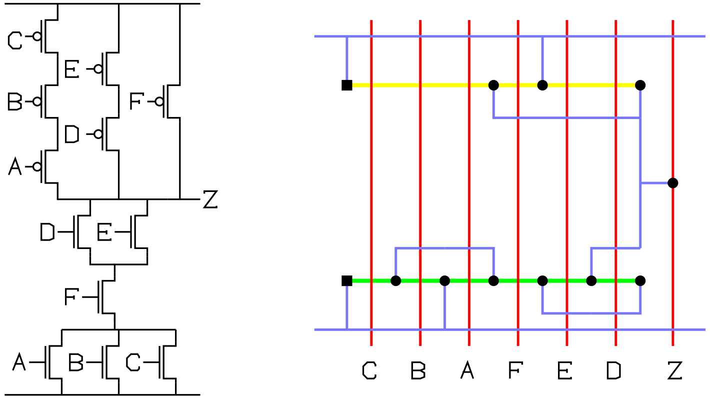
$Z = \overline{(A+B+C) \bullet (D+E) \bullet F}$



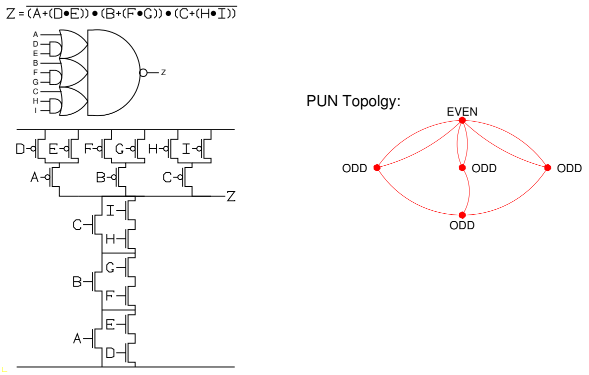
No possible path through n-transistors!

Finding an Euler Path

$Z = \overline{(A+B+C) \bullet (D+E) \bullet F}$

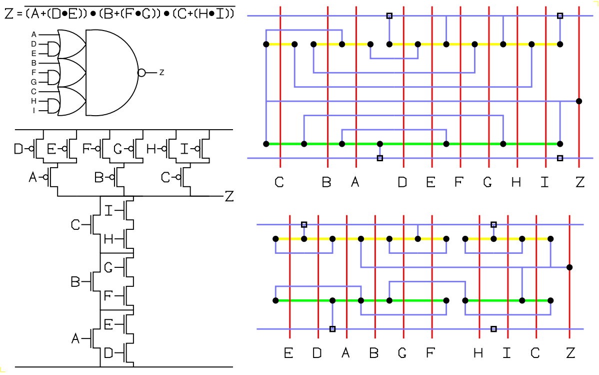


Finding an Euler Path



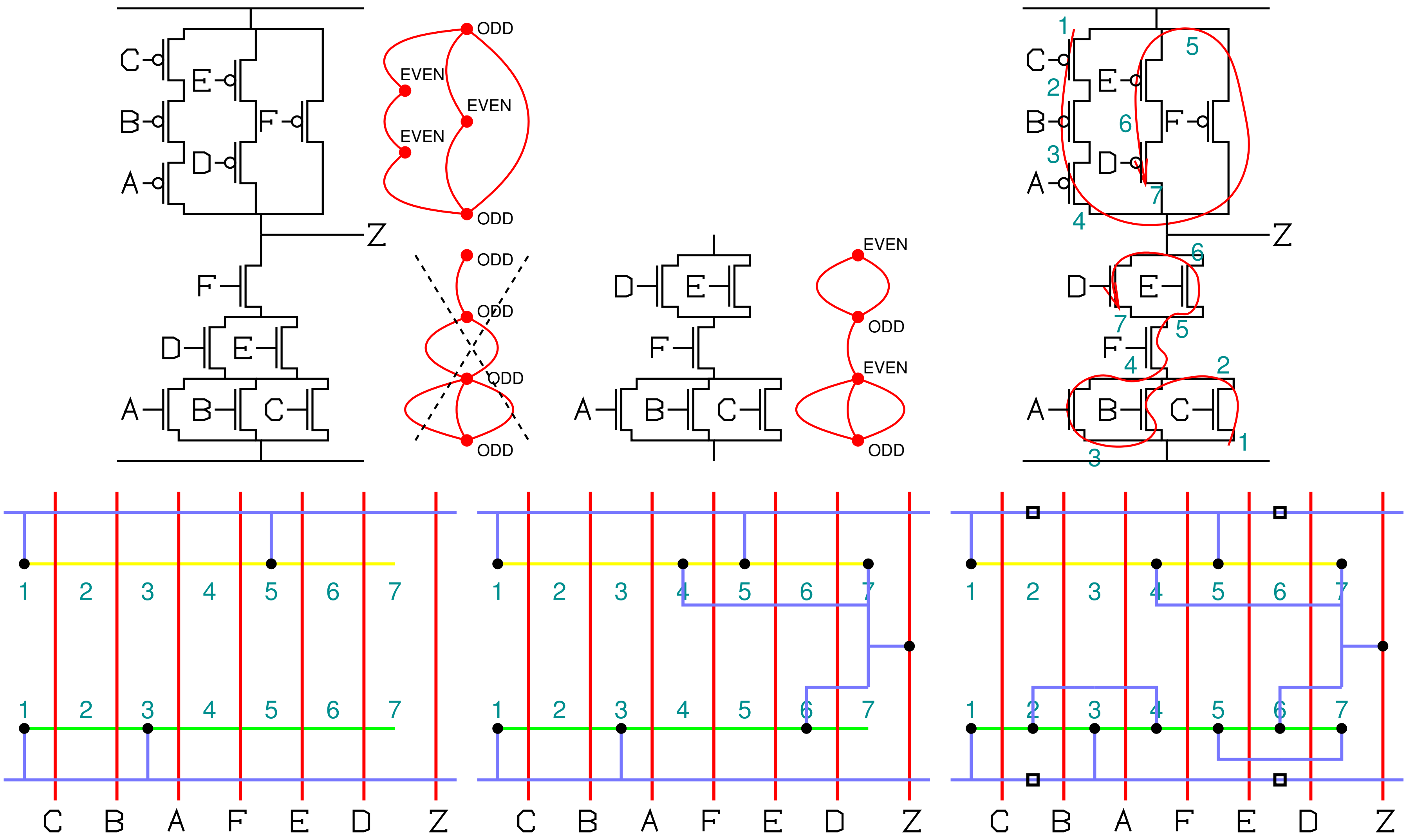
- No possible path through p-transistors.
- No re-arrangement will create a solution!

Philosophers vs. Engineers



- The philosopher is happy to prove that there is no Euler path to be found.
- The engineer will use *partial Euler paths* to reach the best solution.

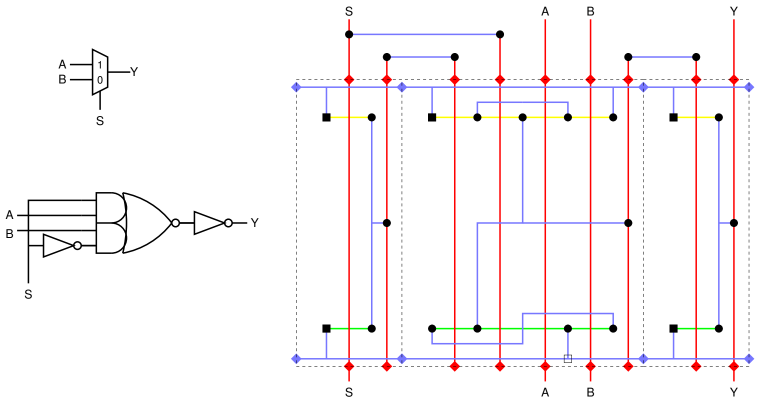
Investigation of Euler paths leads to more efficient layout*



*not all gates will support a common Euler path for both PMOS and NMOS

Digital CMOS Design

Multiple gates



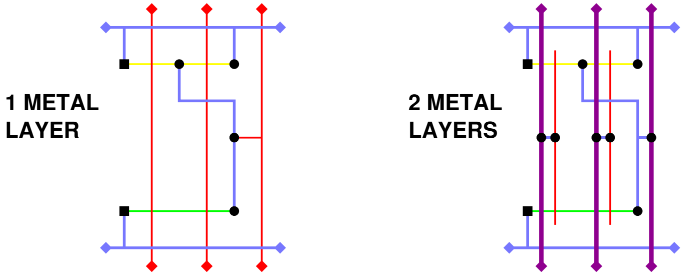
Digital CMOS Design

Multiple gates

- Gates should all be of same height.
  - Power and ground rails will then line up when butted.
- All gate inputs and outputs are available at top and bottom.
  - All routing is external to cells.
  - Preserves the benefits of hierarchy.
- Interconnect is via *two conductor routing*.
  - In this case Polysilicon vertically and Metal horizontally.

Digital CMOS Design

Two-layer Metal



Most modern VLSI processes support two or more metal layers.
The norm is to use only metal for inter-cell routing.

usually	Metal1	for	horizontal	inter-cell routing	(and for power rails)
	Metal2		vertical		(and for cell inputs and outputs).

Standard Cell Design

Many ICs are designed using the standard cell method.

- Cell Library Creation

  A cell library, containing commonly used logic gates[1] is created for a process. This is often carried out by or on behalf of the foundry.

- ASIC[2] Design

  The ASIC designer must design a circuit using the logic gates available in the library.

  The ASIC designer usually has no access to the full layout of the standard cells and doesn't create any new cells for the library.

  Layout work performed by the ASIC designer is divided into two stages:

  - Placement
  - Routing

[1] note that a standard cell may include transistors from more than one basic function (e.g. NAND + inverter to give AND) but will normally be designed *flat* i.e. without layout hierarchy.

[2] Application Specific Integrated Circuit

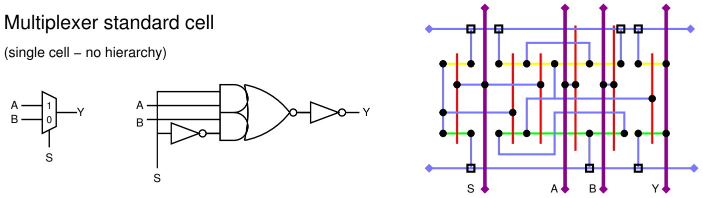
Standard Cell Design

Choosing a set of cells for a cell library

- There is no set size for a cell library.
- Theoretically just one cell ( ) or one type of cell (, , , ...) is sufficient for a cell library.
- The use of more complex cells allows for designs optimised for area and/or performance:

Multiplexer standard cell

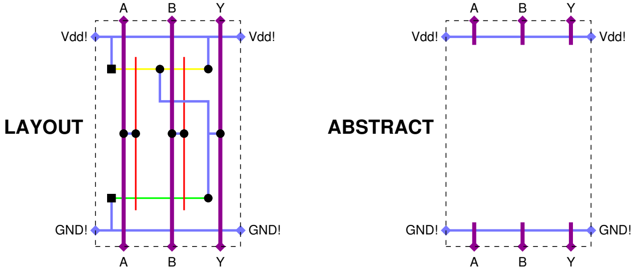
(single cell – no hierarchy)



- Which basic gates; which compound gates; which sequential gates?
- Do we provide different versions of the same gate (e.g. small area version, high drive version)? If so, how many different versions?

Standard Cell Design

Layout and Abstract Views of a Standard Cell

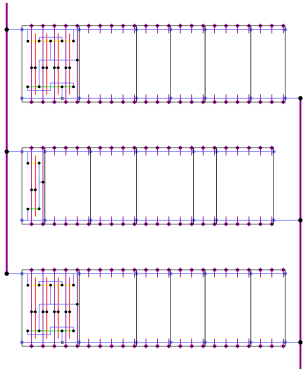


The partial cell layout usually given to the ASIC designer is known as a black box or *abstract* view. The abstract:

- must include cell ports and a cell boundary
- may include some or all of the metal mask information

Placement & Routing

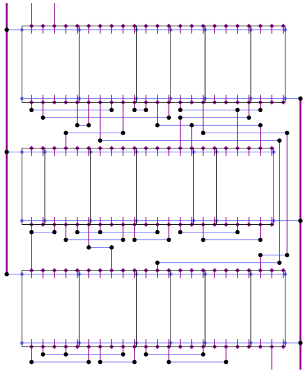
Placement



Cells are placed in one or several equal length lines with inter-digitated power and ground rails.

Placement & Routing

Routing



In the routing channels between the cells we route metal1 horizontally and metal2 vertically.

Placement & Routing

Two conductor routing

Conductor A	for	horizontal	inter-cell routing[3]
Conductor B		vertical	
- This logical approach means that we should never have to worry about signals crossing.

  This makes life considerably easier for a computer (or even a human) to complete the routing.
- We must only ensure that two signals will not meet in the same horizontal or vertical channel.
- Computer algorithms can be used to ensure placement of cells such that wires are short.[4]
- Further computer algorithms can be used to optimize the routing itself.

[3] In the two-metal example Conductor A is Metal1 and Conductor B is Metal2

[4] In VLSI circuits we often find that inter-cell wiring occupies more area than the cells themselves.

Standard Cell Design

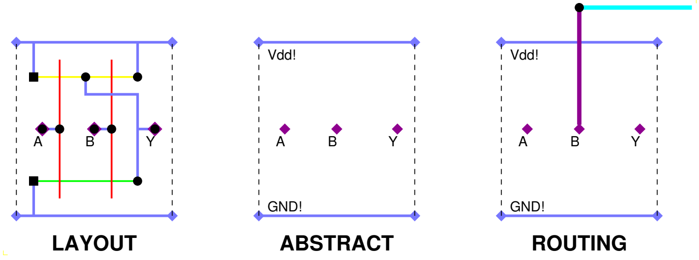
More Metal Layers

With three or more metal layers it is possible to take a different approach.
The simplest example uses three metal layers.

- Standard Cells

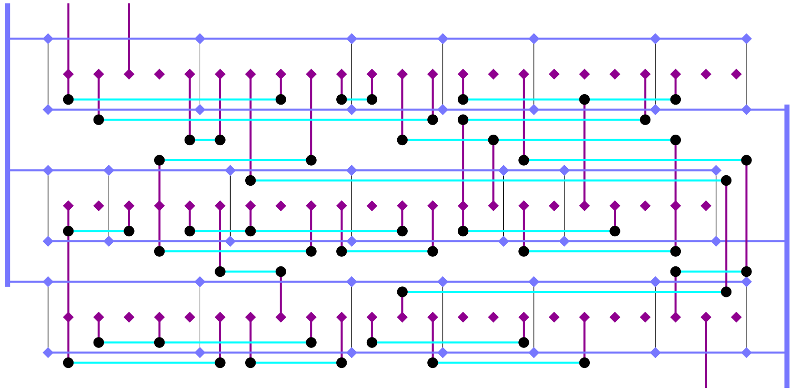
  Use only metal1 except for I/O which is in metal2
- Two Conductor Routing

  Uses metal2 and metal3



Standard Cell Design

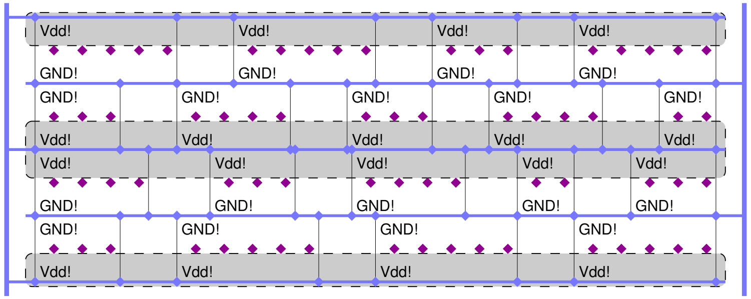
More Metal Layers



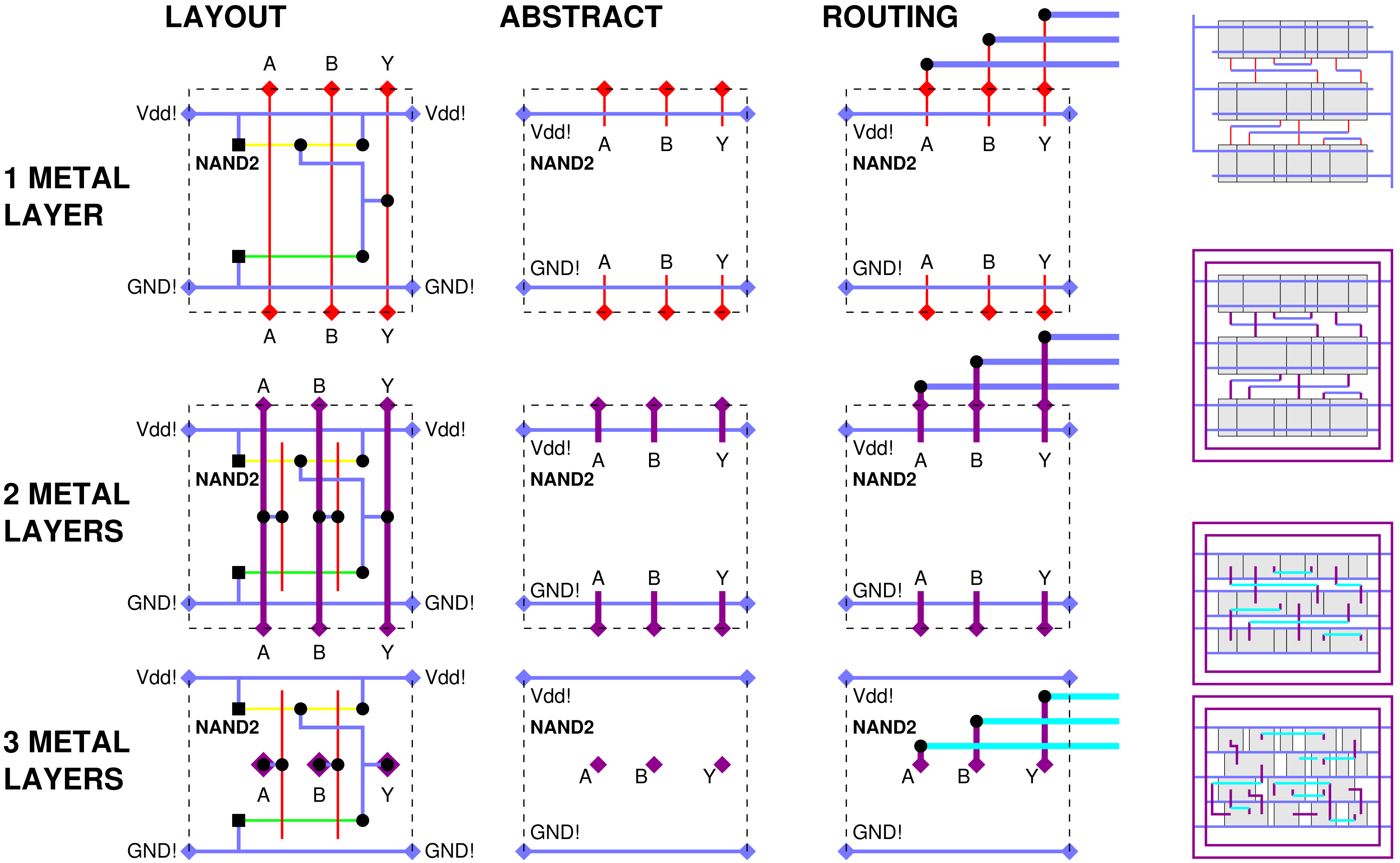
With this approach we can route safely over the cell to the specified pins leading to much smaller gaps between cell rows.

Standard Cell Design

Alternative Placement Style



By flipping every second row it may be possible to eliminate gaps between rows.
N-wells are merged and power or ground rails are shared.
This approach is normally associated with sparse rows and non channel based routing algorithms.



LAYOUT

ABSTRACT

ROUTING

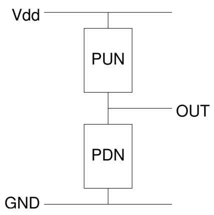
1 METAL LAYER

2 METAL LAYERS

3 METAL LAYERS

7000

Static CMOS Complementary Gates



- **Static**

  After the appropriate propagation delay the ouput becomes valid and remains valid.[1]

- **Complementary**

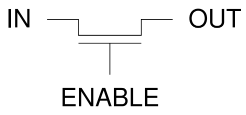
  For any set of inputs there will exist either a path to Vdd or a path to GND.

  Where this condition is not met we have either a high impedence output or a conflict in which the strongest path succeeds. Static CMOS **Non-complementary** gates make use of these possibilities.

[1] c.f. Dynamic logic which uses circuit capicitance to store state for a short time.

Pass Transistor Circuits

- Pass Transistor



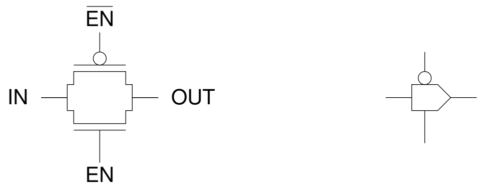
  - Provides very compact circuits.
  - Good transmission of logic '0'.
  - Poor transmission of logic '1'.
    - - slow rise time
    - - degradation of logic value

The pass transistor is used in many dynamic CMOS circuits[2].

[2] where pull-up is performed by an alternative method

Pass Transistor Circuits

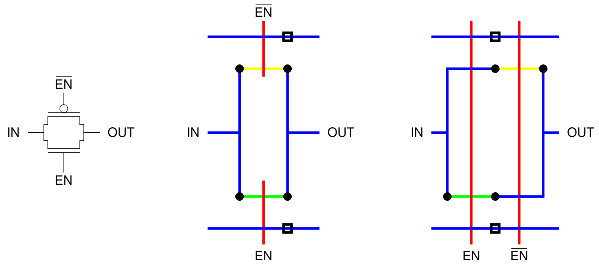
- Transmission Gate
  - For static circuits we would normally use a CMOS transmission gates:



- - balanced *n* and *p* pass transistors
- - faster pull-up
- - slower pull-down

Pass Transistor Circuits

- Transmission Gate Layout

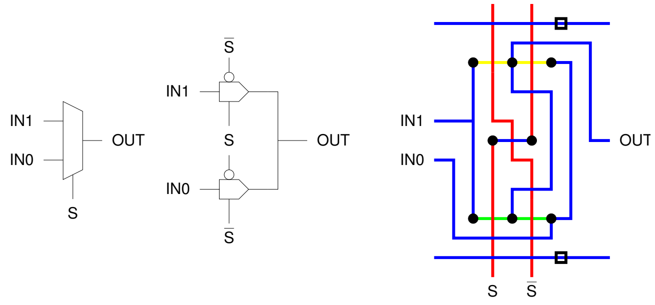


  - note that these circuits are not fully complementary[3] hence they do not immediately lend themselves to a *line of diffusion* implementation.

[3] since there are sets of inputs for which the output is neither pulled low nor high

Pass Transistor Circuits

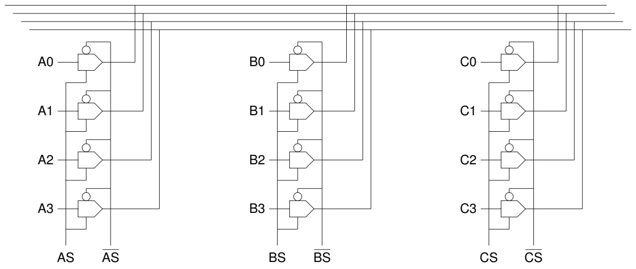
- Transmission Gate Multiplexor



  - very few transistors 4 (+2 for inverter)
  - difficult layout may offset this advantage
    - - prime candidate for 2 level metal

Pass Transistor Circuits

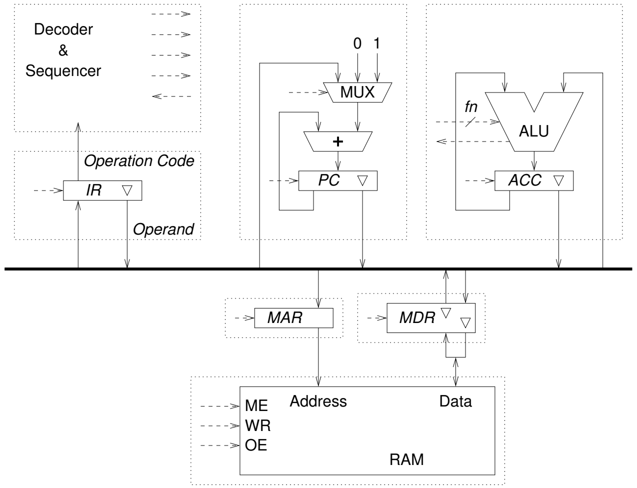
- Bus Wiring



  - distributed multiplexing[4]
  - only one inverter required per bank of transmission gates
  - greatly simplifies global wiring

[4] internal chip bus should never be allowed to float high impedance

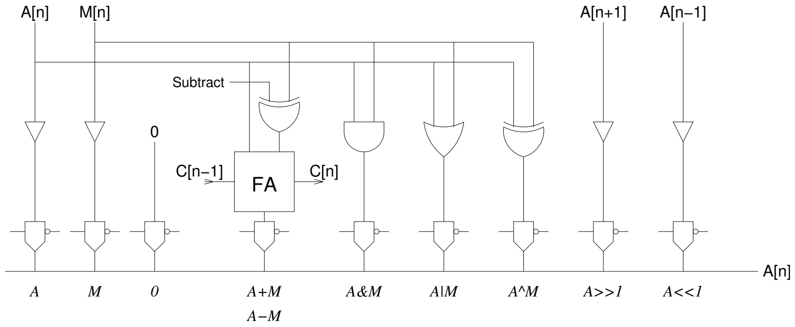
Bus Distributed Multiplexing



Ideal for signals with many drivers from different modules.

Bus Distributed Multiplexing

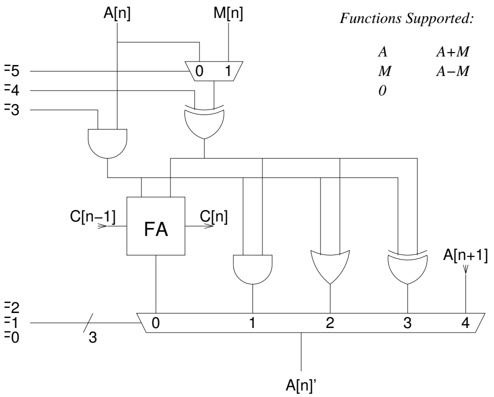
Implementation of bitslice ALU:[5]



- Separate circuit for each function
- Connected via distributed multiplexor

[5] Note that transmission gates have no drive capability in themselves. Here a good drive is ensured by providing buffers.

Bus Distributed Multiplexing



Functions Supported:

A
M
0

A+M
A−M

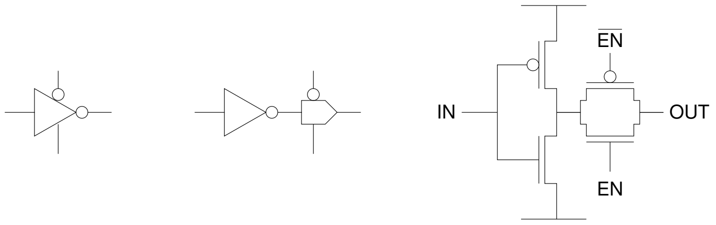
A>>1
A<<1

~A
~M
A&M
A|M
A^M
~(A^M
A&~M
A|~M

- Single optimized ALU module
- Multiplexing is not distributed
- Multiplexor implementation may use transmission gates

Pass Transistor Circuits

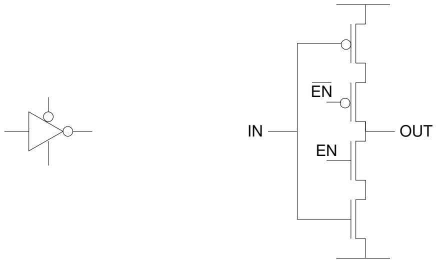
- Tristate Inverter



  - Any gate may have a tri-state output by combining it with a transmission gate.

Pass Transistor Circuits

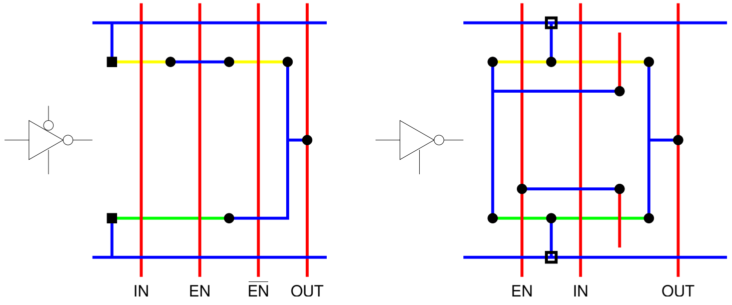
- Tristate Inverter



  - Alternatively the transmission gate may be incorporated into the gate.
    - - - one connection is removed - easier to layout
    - - - also easier to simulate!

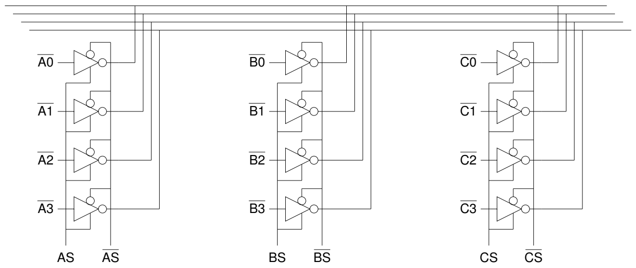
Pass Transistor Circuits

- Tristate Inverter Layout



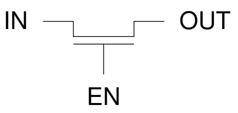
Pass Transistor Circuits

- Tristate Inverter Bus Driver

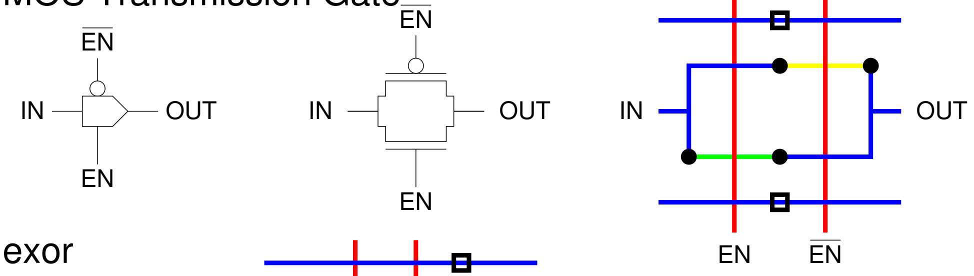


  - a tristate inverting buffer is often used to drive high capacitance bus signals
  - transistors may be sized as required

Pass Transistor

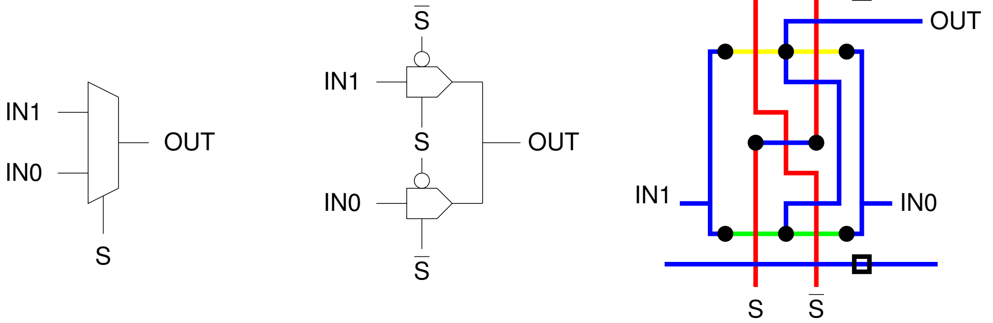


CMOS Transmission Gate



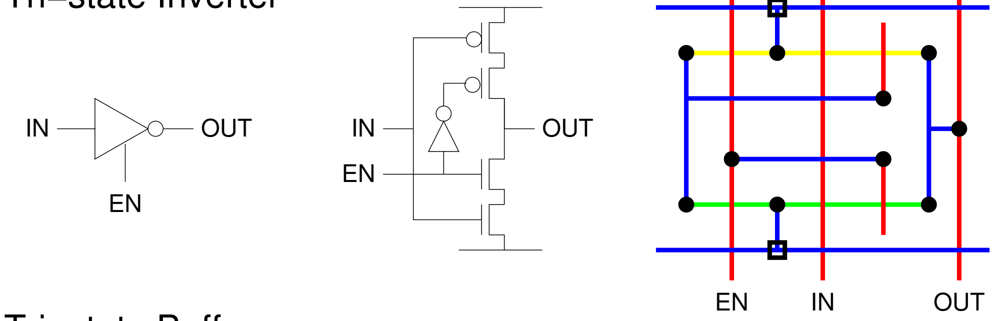
Tri–state gates are used for Multiplexing

Transmission Gate Multiplexor

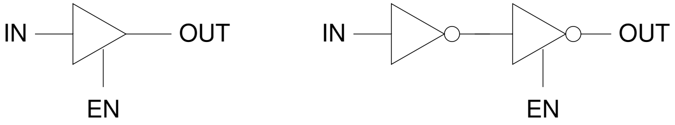


*note distictive polysilicon crossover

Tri–state Inverter



Tri–state Buffer

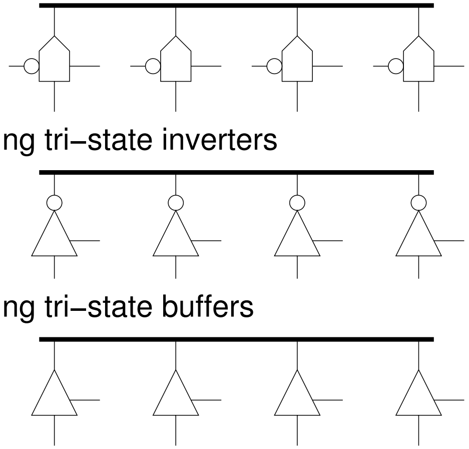


Distributed Multiplexing

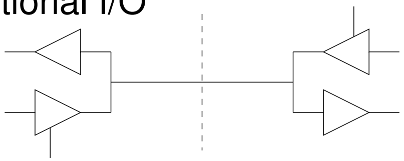
using transmission gates

using tri–state inverters

using tri–state buffers



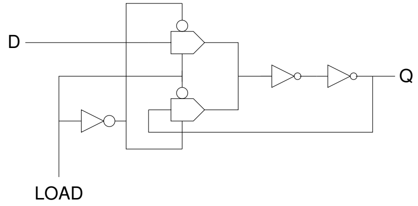
Bi–directional I/O

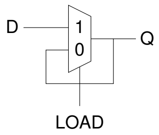


*this is another form of multiplexing

Latches and Flip-Flops

- CMOS transmission gate latch



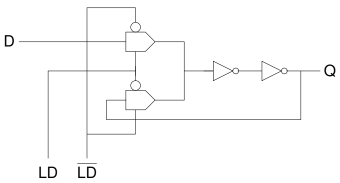


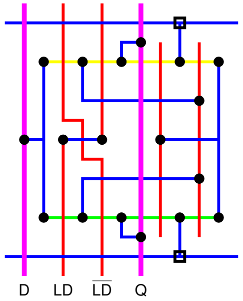
A simple transparent latch can be build around a transmission gate multiplexor

- transparent when load is high
- latched when load is low
- two inverters are required since the transmission gate cannot drive itself

Latches and Flip-Flops

- Transmission gate latch layout

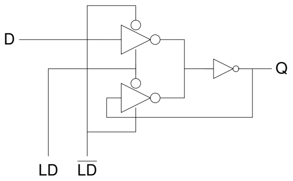


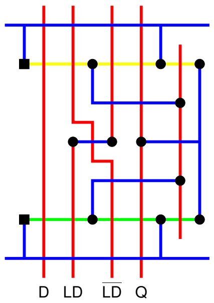


- a compact layout is possible using 2 layer metal

Latches and Flip-Flops

- A simpler layout may be achieved using tristate inverters.

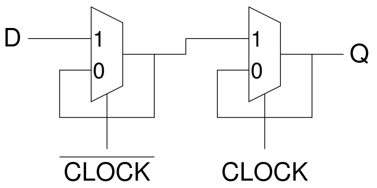




- this design requires two additional transistors but may well be more compact.

Latches and Flip-Flops

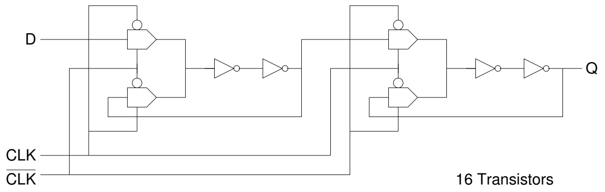
- For use in simple synchronous circuits we use a pair of latches in a master slave configuration.



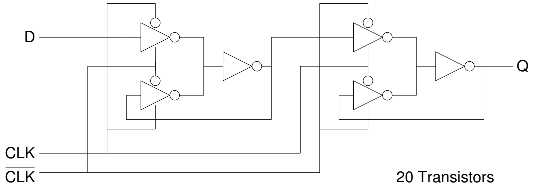
- this avoids the race condition in which a transparent latch drives a second transparent latch operating on the same clock phase.
- the circuit behaves as a rising edge triggered D type flip-flop.

Latches and Flip-Flops

- Transmission gate implementation

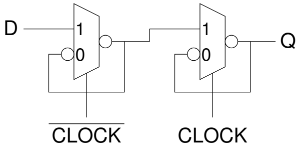


- Tristate inverter implementation

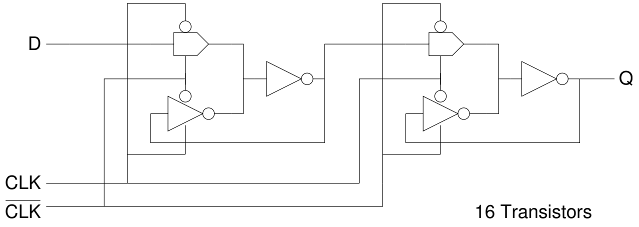


Latches and Flip-Flops

- Alternative configuration

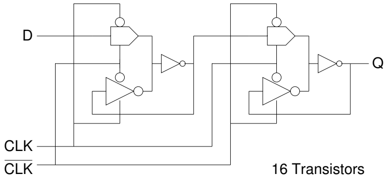


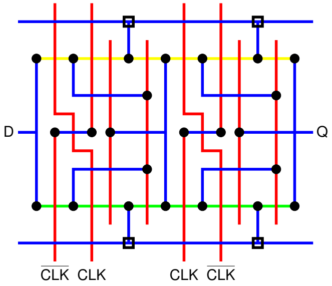
  - Implementation



Latches and Flip-Flops

- Layout of master slave D type.

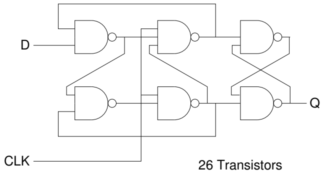


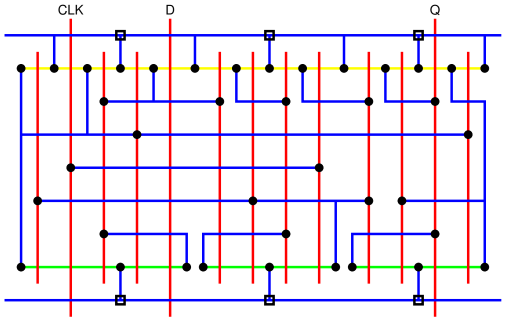


  - very compact using alternative configuration.

Latches and Flip-Flops

- For the same functionality we could use an edge triggered D type:

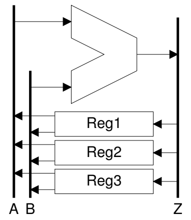


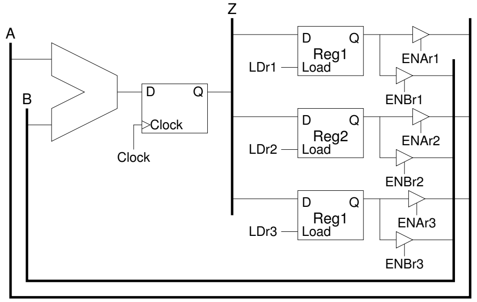


  - a few more transistors
  - more complex wiring
  - simpler clock distribution

Register File

Where we have large amounts of storage the use of individual latches can lead to space saving.

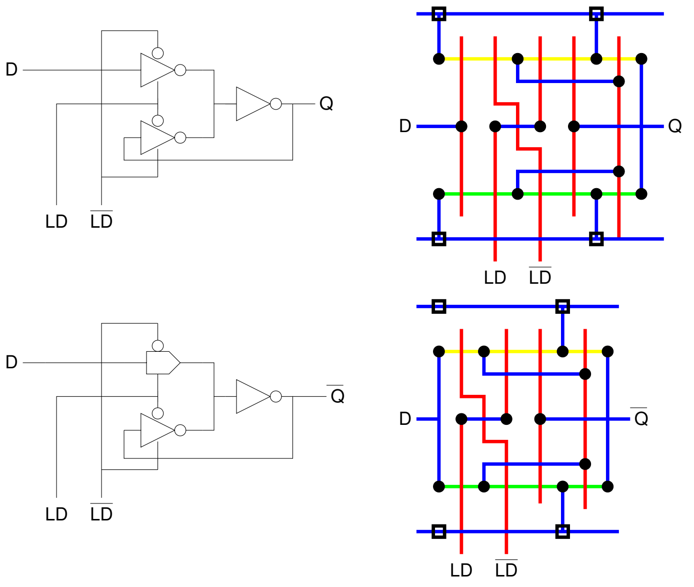


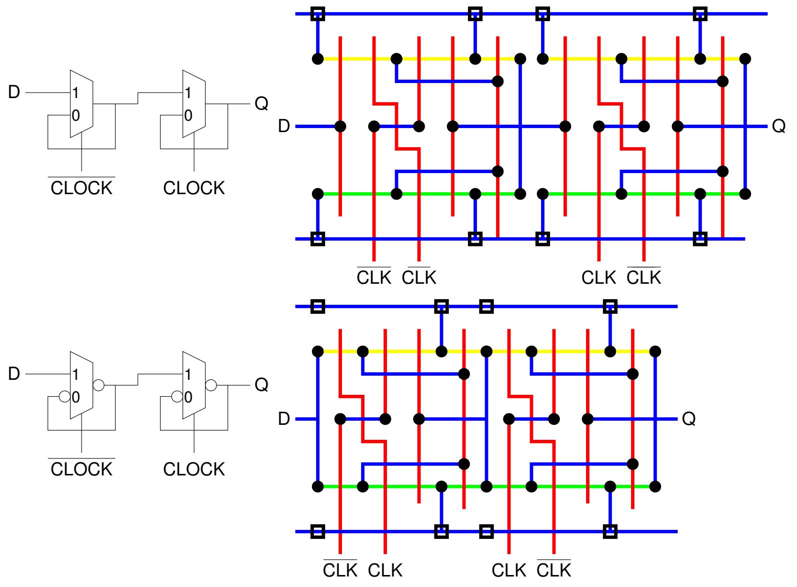


- Load signals must be glitch free with tightly controlled timing.
- Edge Triggered D-type prevents a race condition ($Reg1 \leftarrow Reg1 + Reg2$).

Latch

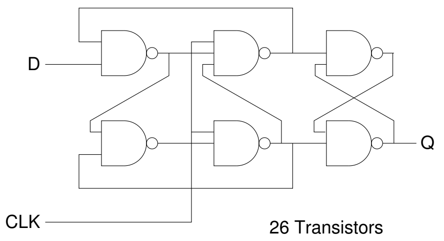
Master Slave D−Type Flip−Flop

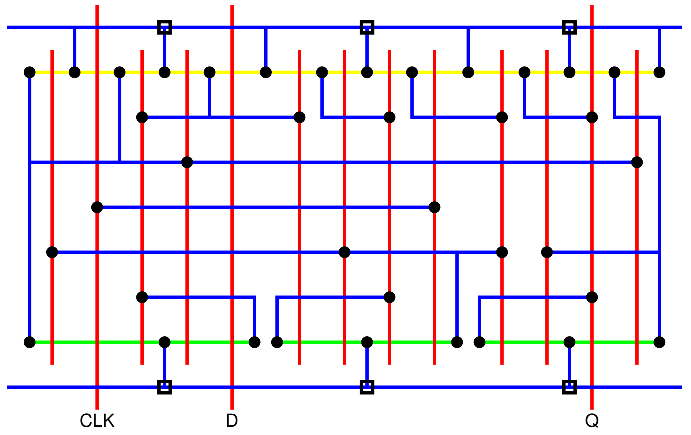




*muliplexor based latches and flip−flops include distictive polysilicon crossover

Edge Triggered D−Type Flip−Flop

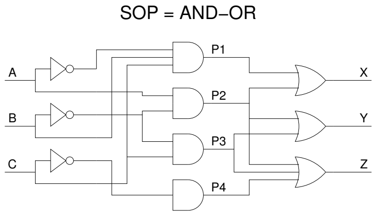


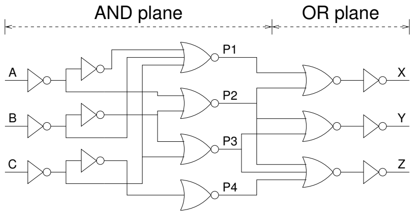


Euler path analysis is applied creatively to these multi−gate cells – gates are often linked via the common gnd/pwr node
Final layouts will be more complex where clock buffers, reset circuitry and metal 2 i/o are included

PLA structures

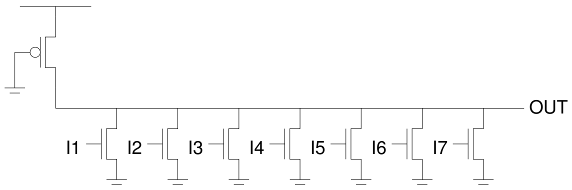
Programmable Logic Array structures provide a logical and compact method of implementing multiple SOP (Sum of Products) or POS expressions.





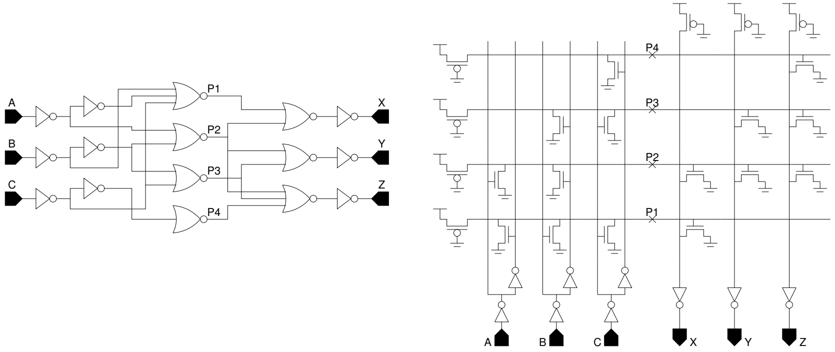
Most PLA structures employ pseudo-NMOS NOR gates using a P-channel device in place of the NMOS depletion load.

Pseudo-NMOS NOR gate



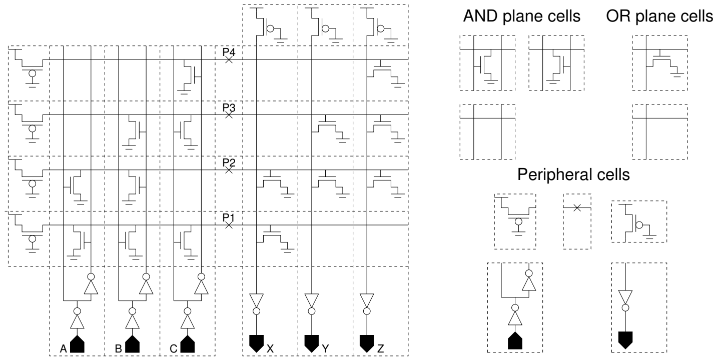
- Unlike complementary CMOS circuits, these gates will dissipate power under static conditions (since the P device is always on).
- The P and N channel devices must be ratioed in order to create the required low output voltage.
- This ratioing results in a slower gate, although there is a trade-off between gate speed and static power dissipation.

PLA structure



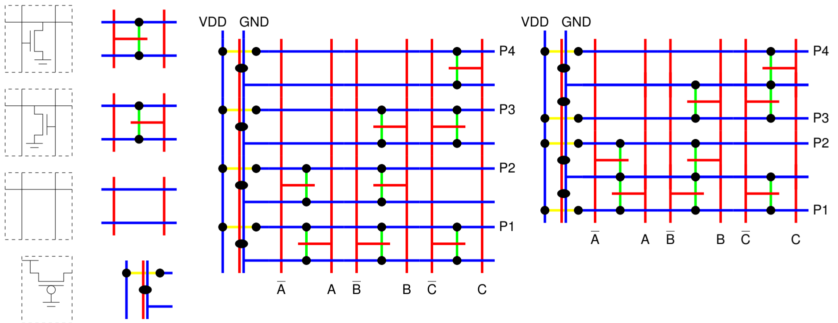
- A regular layout is employed, with columns for inputs and outputs and rows for intermediate expressions.

PLA structure



- Layout is simply a matter of selecting and placing rectangular cells from a limited set.

PLA structure

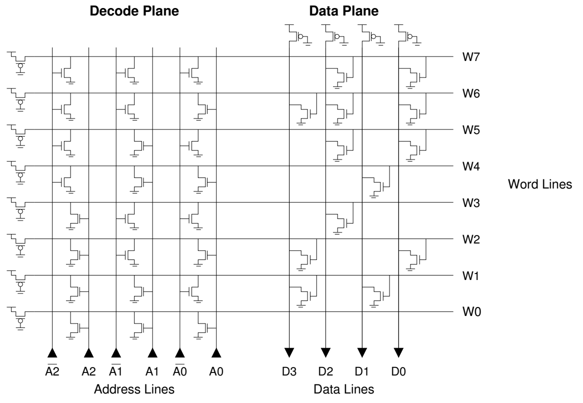


Only AND plane cells are shown here

- Conversion to *sticks* is straight forward with opportunities for further optimization.

ROMs

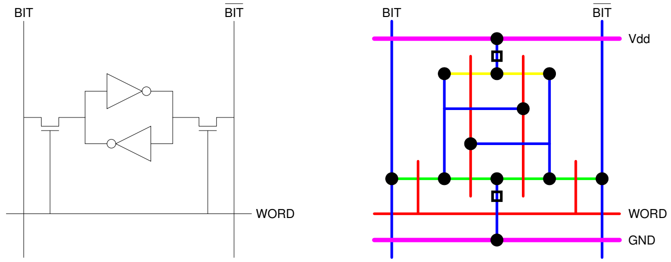
- A ROM may simply be a PLA with fixed decode plane[1] and programmable data plane.



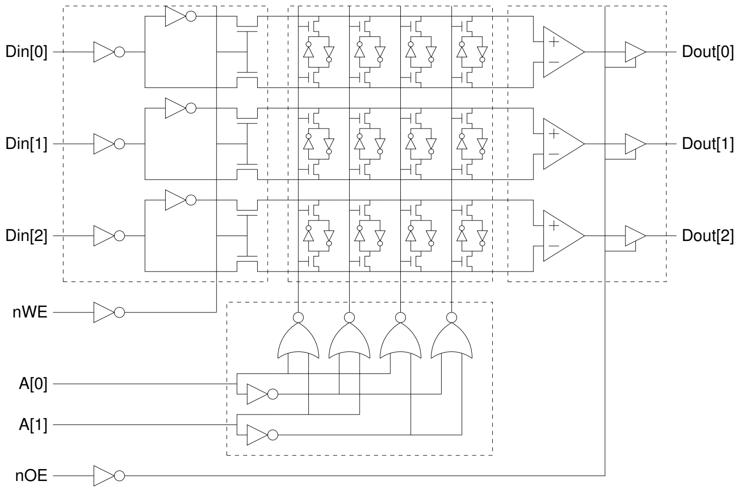
[1] RAM structures can make use of the same decode plane.

Static RAM

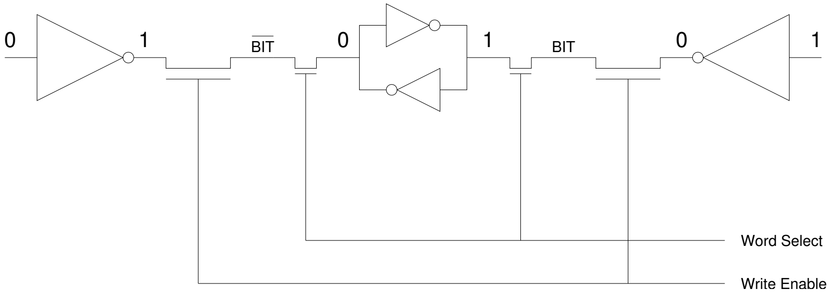
- Used for high density storage on a standard CMOS process.
- Short lived conflict during write - NMOS transistors offer stronger path.
- Differential amplifiers are used for speedy read.



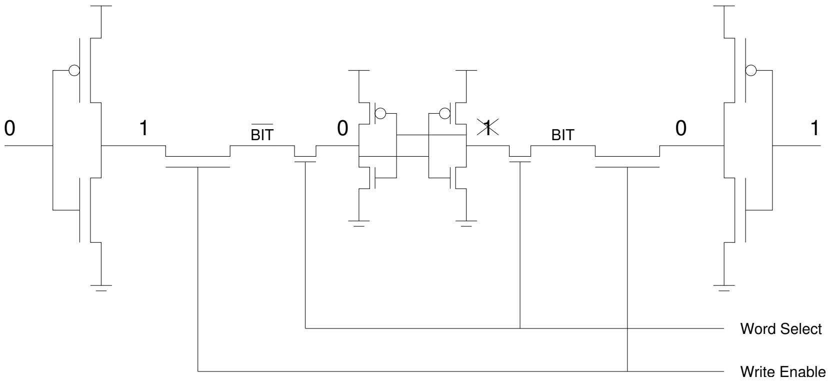
Standard 6 transistor static RAM cell.



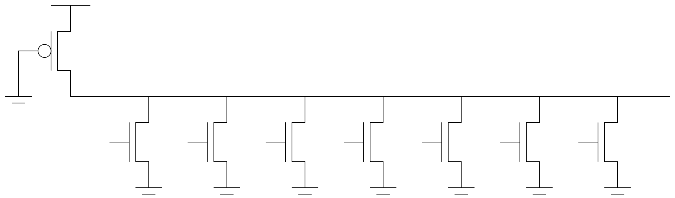
SRAM Write

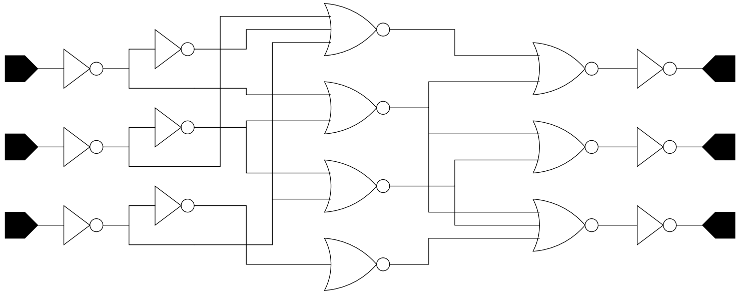


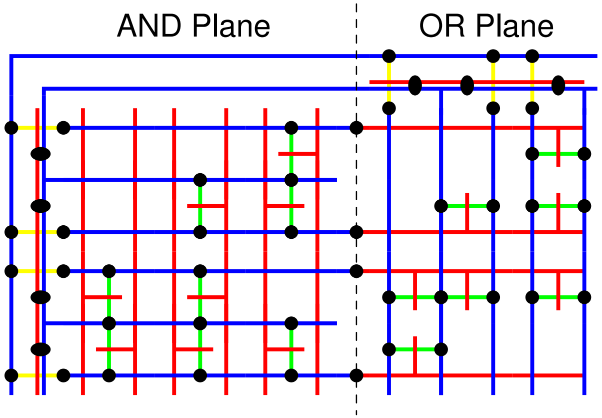
SRAM Write



PLA and ROM



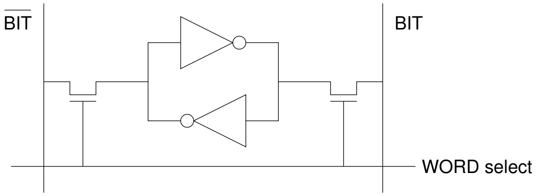


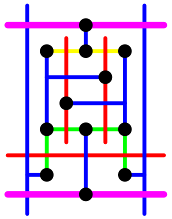


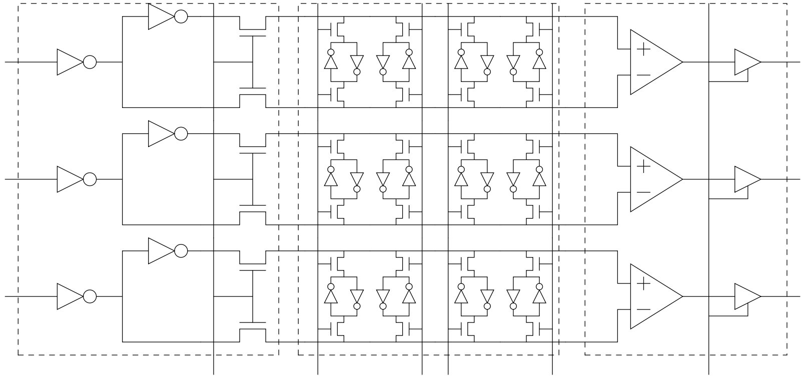
ROM is PLA with

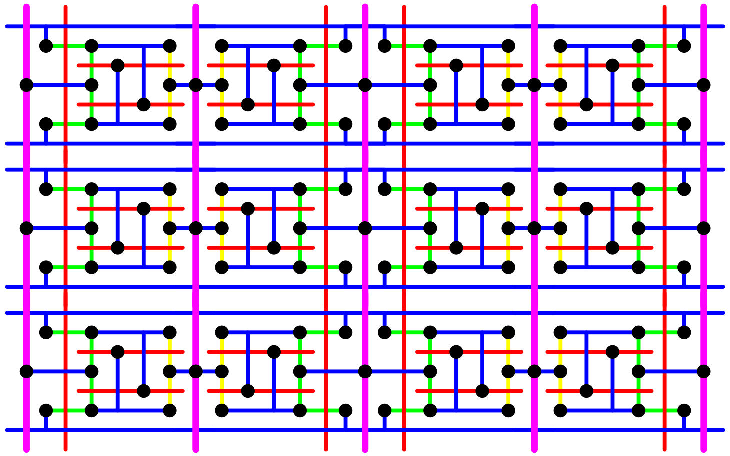
fixed AND (decoder) plane
programmable OR (data) plane

SRAM









Cells are designed to butt together in two dimensions leading to efficient layout

PLA layout efficiency will depend on the actual function implemented (e.g. number of common product terms)

System Design Choices

- Programmable Logic
  - PLD
    e.g. Lattice ispGAL22V10, Atmel ATF1502 CPLD
  - Field Programmable Gate Array (FPGA)
    e.g. Intel Cyclone V, Xilinx Artix-7/Zync-7000
- Semi-Custom Design
  - Mask Programmable Gate Array
    e.g. ECS CMOS Gate Array
    Intel HardCopy II structured ASICs
  - Standard Cell Design
    e.g. AMS CORELIB 0.35μm cell library
- Full Custom Design

System Design Choices

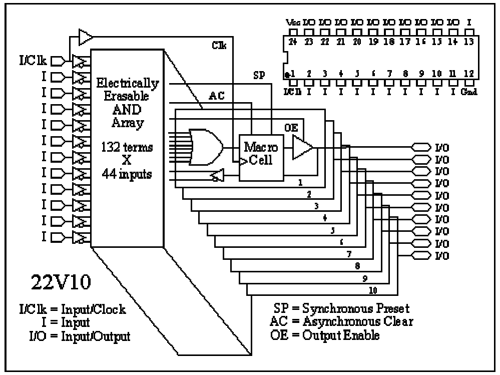
- Programmable Logic

START HERE

  - Best possible design turnaround time
  - Cheapest for prototyping
  - Best time to market
  - Minimum skill required
- Semi-Custom Design
- Full Custom Design
  - Cheapest for mass production
  - Fastest
  - Lowest Power
  - Highest Density[1]
  - Most skill required

[1] optimization limited by speed/power/area trade off

Programmable Logic

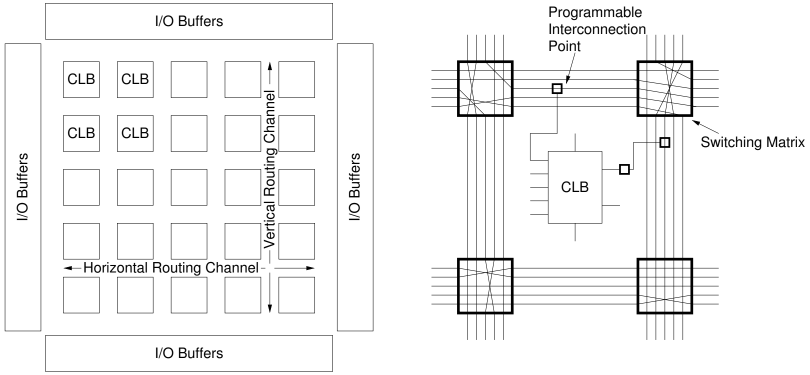


ICT PEEL22CV10

Source: ICT

- One time use - Fuse programmable.
- Reprogrammable - UV/Electrically Erasable.

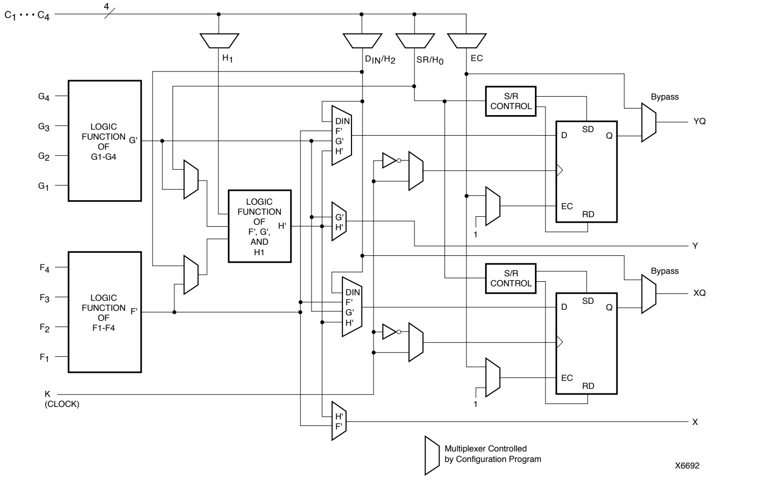
Field Programmable Gate Array – Xilinx XC4000



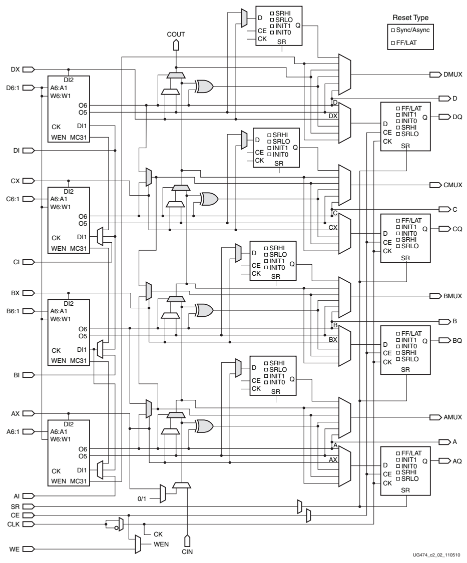
- Configurable Logic Blocks (CLBs) & I/O Blocks[2]
- Programmable Interconnect

[2] Xilinx XC4013 has 576 (24 × 24) CLBs and up to 192 (4 × 48) user I/O pins.

Field Programmable Gate Array – Xilinx XC4000 CLB



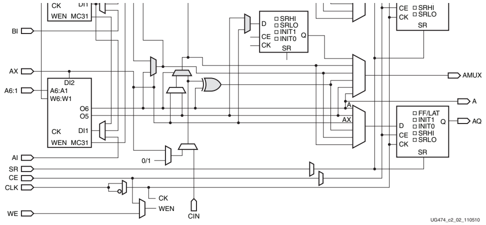
Source: Xilinx



Artix-7 – SLICEM CLB

Source: Xilinx

Xilinx Artix-7 – SLICEM CLB[3]



Source: Xilinx

- 4x 6-input Look-Up Tables (LUTs) for combinational logic
- Carry chain supporting fast carry lookahead
- 8x storage elements

LUTs can be alternatively configured as

- 256 bits RAM
- 32-bit shift register

[3] Xilinx XC7A200T has 16,825 CLBs (each containing 2 slices) and up to 500 user I/O pins.

FPGA - System On Chip

Modern FPGAs are big enough for:

- One or more soft-core processors
- Program memory
- Data memory

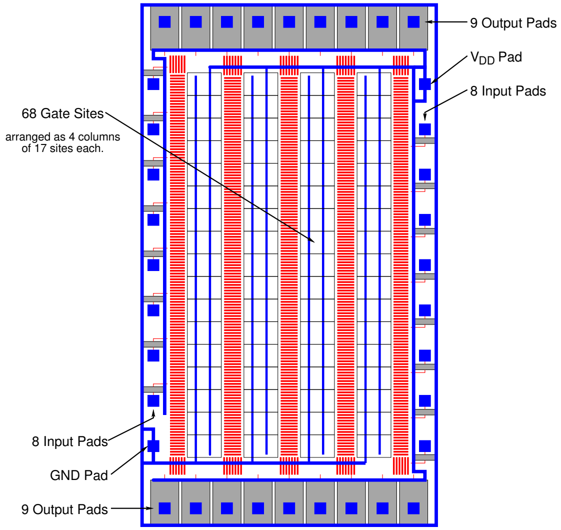
  + specialist hardware

The new trend is for FPGAs with hard processors built in:

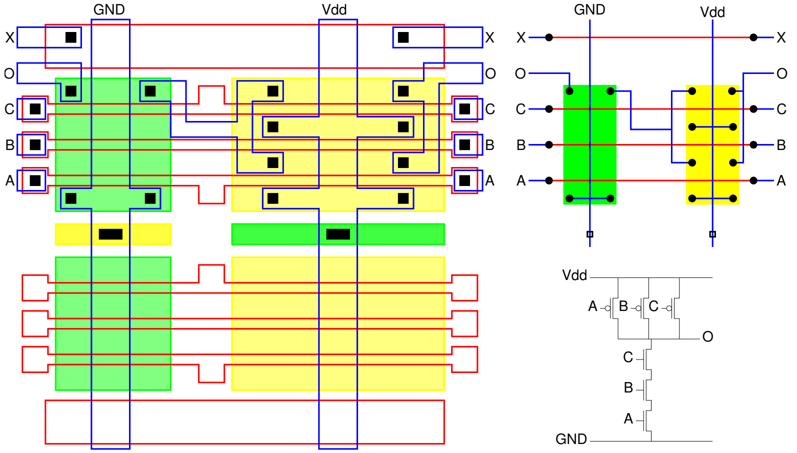
- Xilinx Zync-7000 includes dual-core ARM A9
- Intel Cyclone V SE includes dual-core ARM A9
- Cypress PSoC 4 includes ARM Cortex-M0 and programmable digital[4] and analog blocks

[4] here the digital block is PLD rather than FPGA

Mask Programmable Gate Array



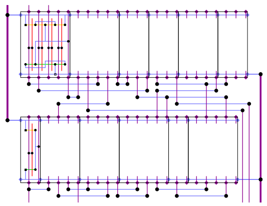
Mask Programmable Gate Array



- Customize Metal and Contact Window masks only.

Standard Cell Design

- Logic Functions



- Auto Generated Macro Blocks
  - PLA
  - ROM
  - RAM
- System Level Blocks
  - Microprocessor core[5]

[5] Will support System On Chip applications.

Full Custom

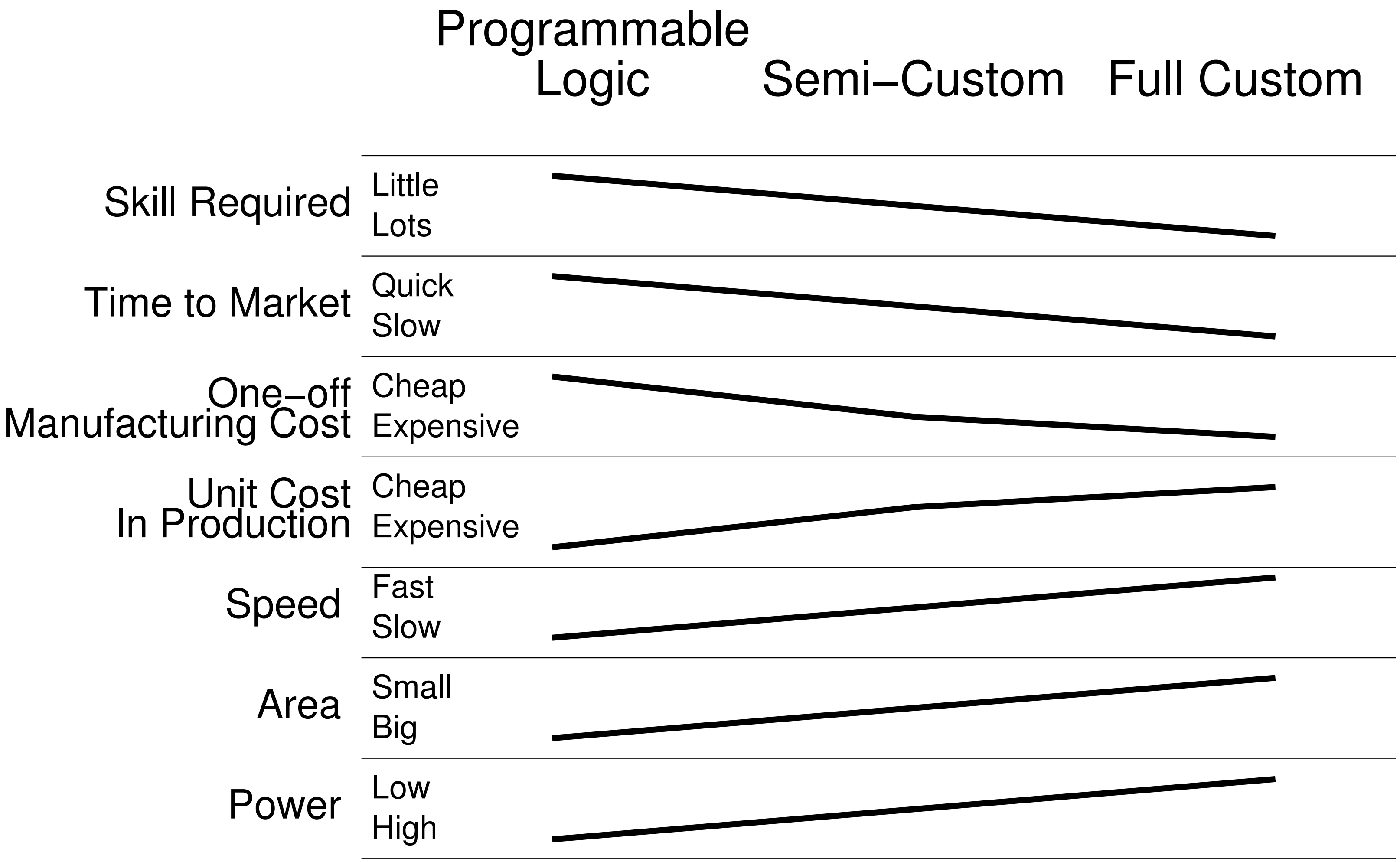
All design styles need full custom designers

- to design the base programmable logic chips
- to design building blocks for semi-custom

Where large ASICs use full custom techniques they are likely to be used alongside semi-custom techniques.

e.g. Hand-held computer game chip

- Full custom bitslice datapath
  hand crafted for optimum area efficiency and low power consumption
- Standard cell controller
- Macro block RAM, ROM



Programmable Logic

Semi–Custom

Full Custom

Skill Required: Little / Lots

Time to Market: Quick / Slow

One–off Manufacturing Cost: Cheap / Expensive

Unit Cost In Production: Cheap / Expensive

Speed: Fast / Slow

Area: Small / Big

Power: Low / High

All design styles need full custom designers

A large ASIC (especially SoC) may mix Semi–Custom and Full Custom