

ENR-325/325L Principles of Digital Electronics and Laboratory

Xiang Li

Fall 2025



COE COLLEGE®

A homemade MOSFET by Sam Zeloof in 2021:

- <https://www.youtube.com/watch?v=IS5ycm7VfXg>

Also check:

<https://sam.zeloof.xyz/category/semiconductor/>

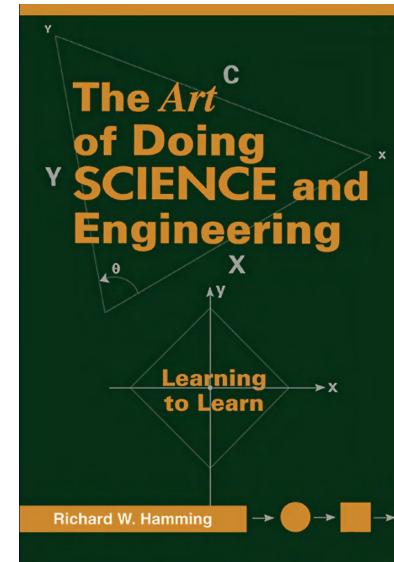


COE COLLEGE®

Making CMOS is in the realm of process engineering

In science if you know what you are doing you should not be doing it.

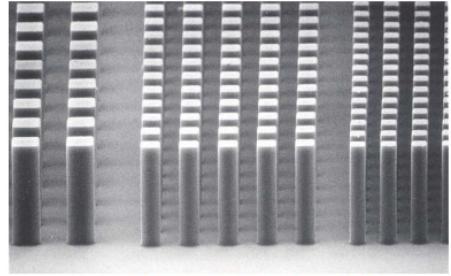
In engineering if you do not know what you are doing you should not be doing it.



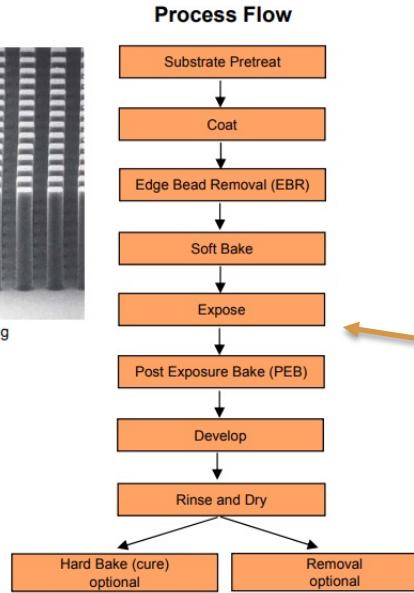
COE COLLEGE[®]

The realm of process engineering

Photolithography example: lift-off



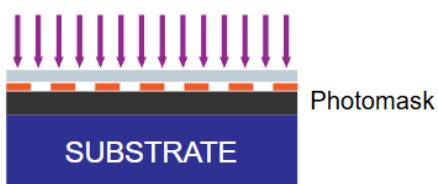
10 um features, 50 um SU-8 2000 coating



1) Spin-coating of photoresist



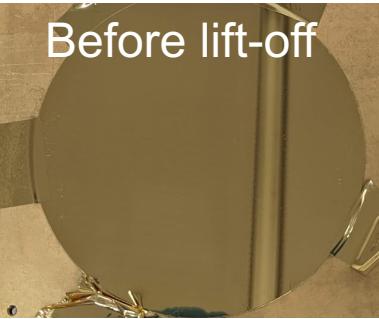
2) UV light



3) Pattern development



Photolithography workflow



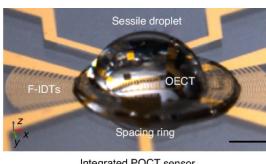
Before lift-off



Lifting-off



Vacuum
chamber



COE COLLEGE®

Data sheet and process flow from Kayaku and Kurt J. Lesker

Fabrication process at Indiana University cleanroom, 2021

Device shown was for Alzheimer's detection: <https://doi.org/10.1038/s41378-025-00916-4>

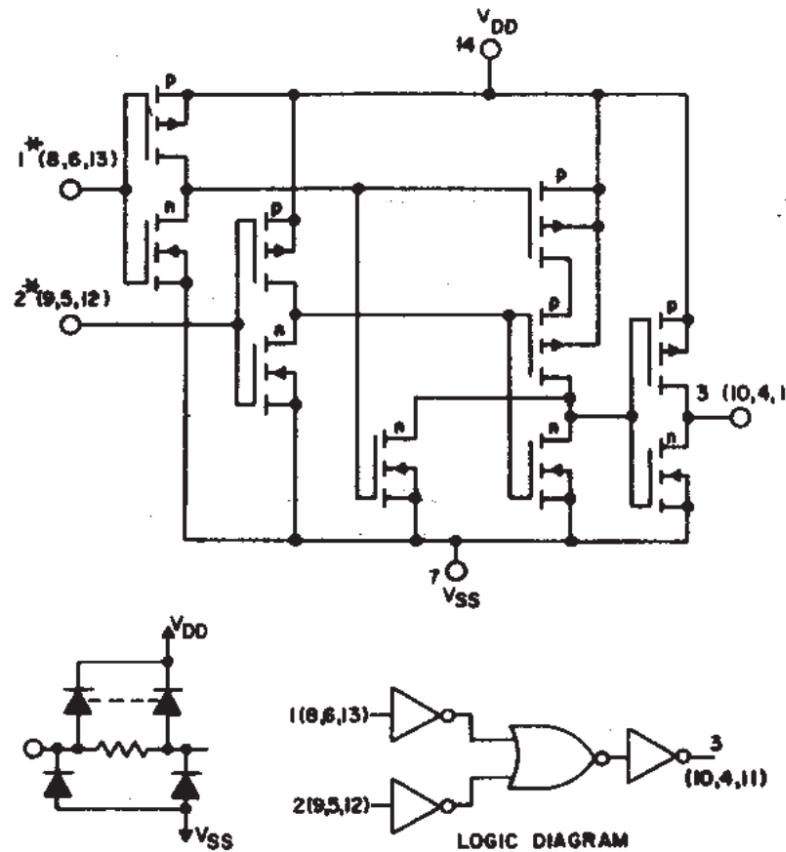
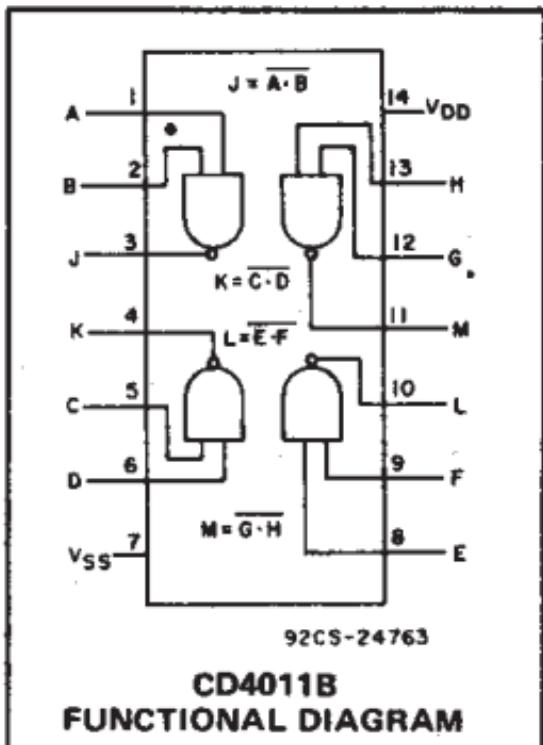
CMOS NAND gate IRL:



TEXAS
INSTRUMENTS

Data sheet acquired from Harris Semiconductor
SCHS021D – Revised September 2003

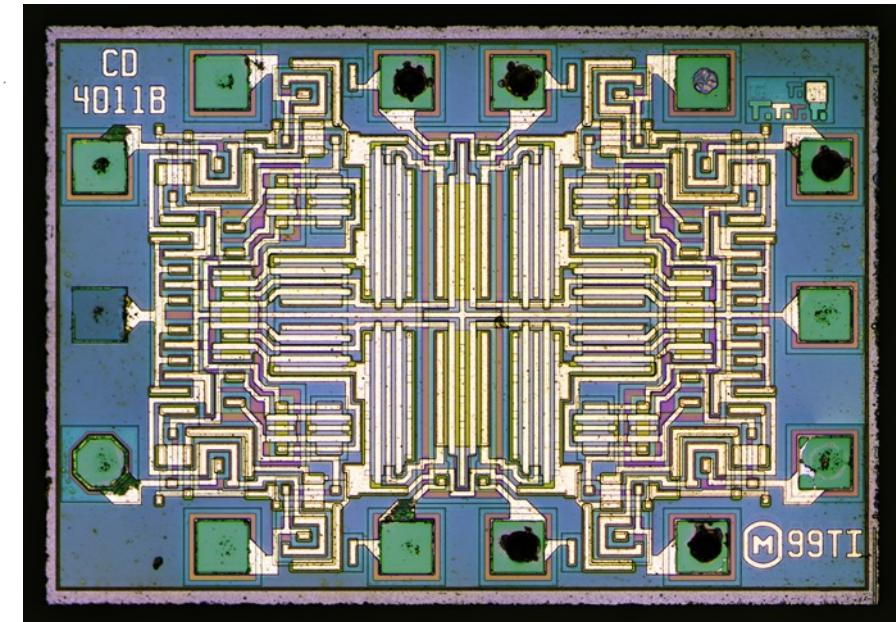
CMOS NAND GATES



* ALL INPUTS ARE PROTECTED
BY CMOS PROTECTION
NETWORK

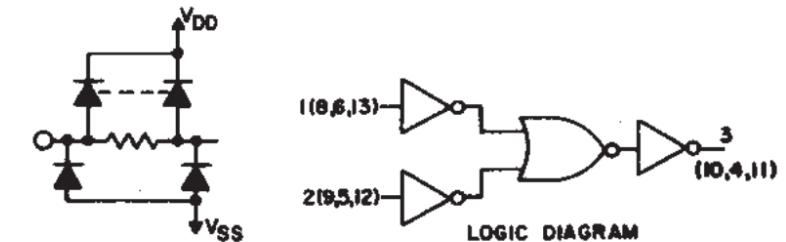
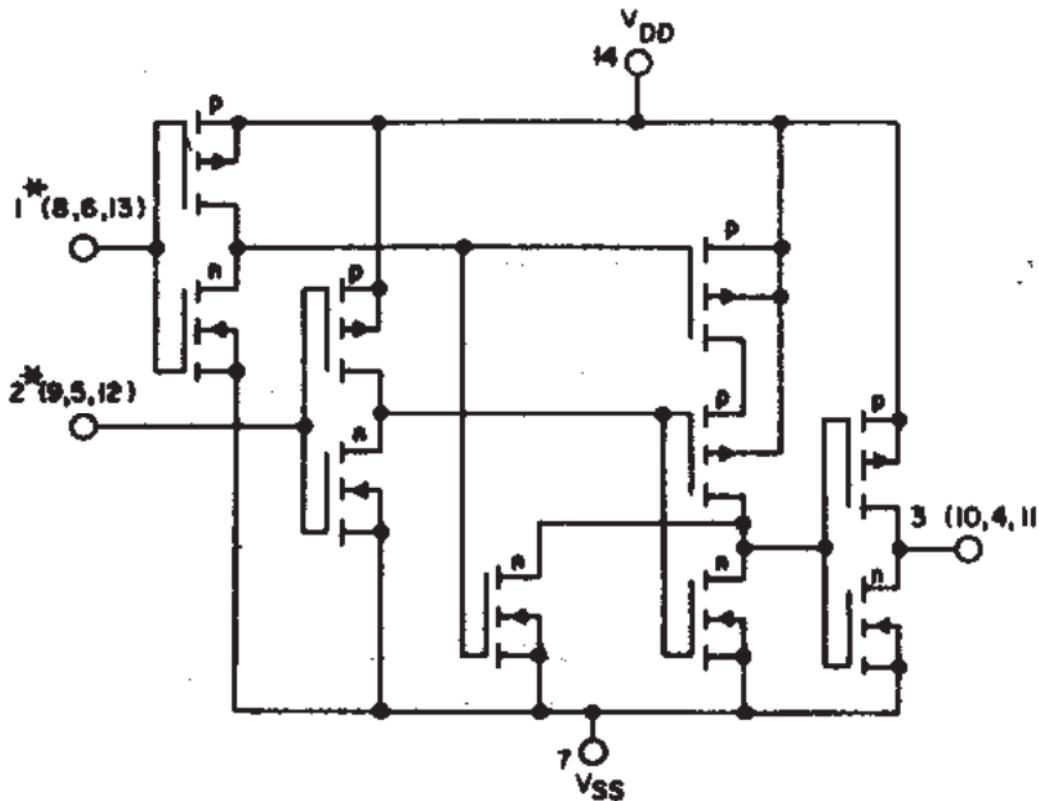
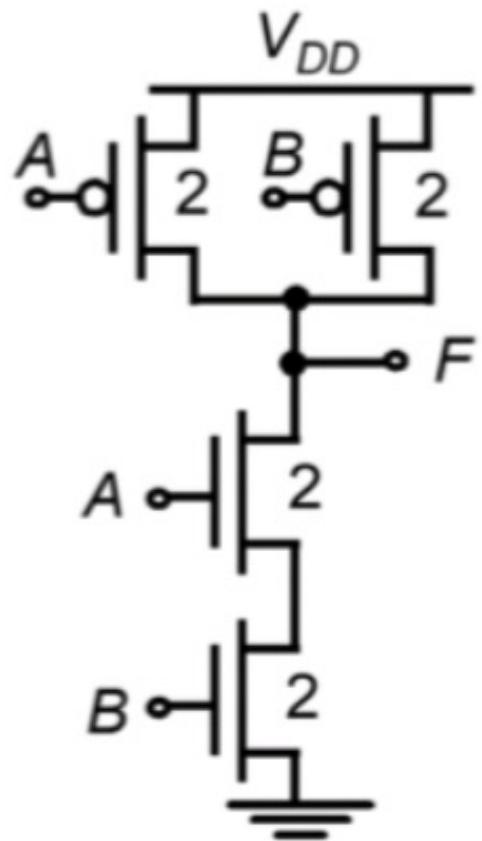
1 OF 4 GATES (NUMBERS
IN PARENTHESES ARE
TERMINAL NUMBERS
FOR OTHER GATES)

Fig. 7 – Schematic and logic diagrams for CD4011B.



COE COLLEGE®

Little class exercise: identify the NAND gate



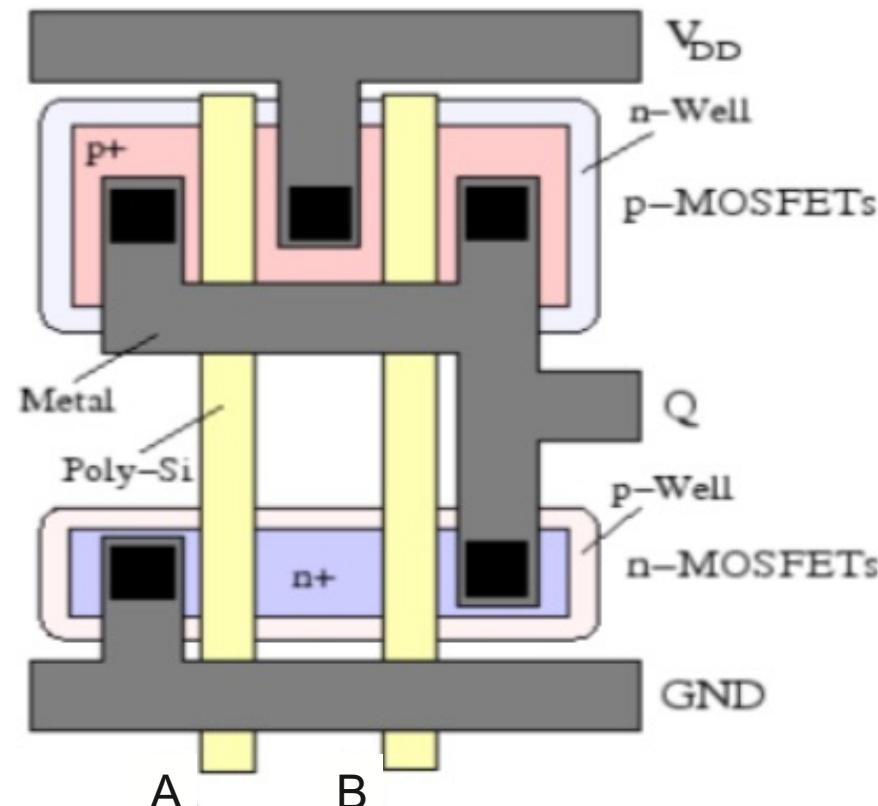
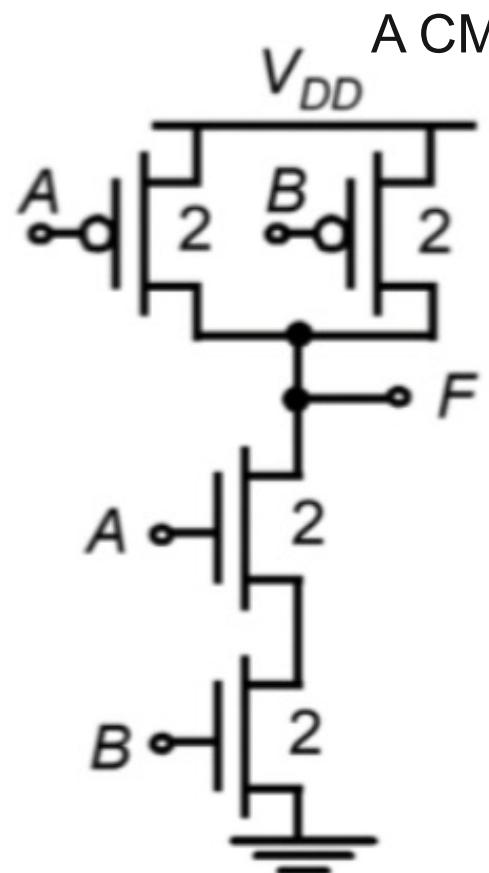
* ALL INPUTS ARE PROTECTED
BY CMOS PROTECTION
NETWORK

I OF 4 GATES (NUMBERS
IN PARENTHESES ARE
TERMINAL NUMBERS
FOR OTHER GATES)

Fig.7 – Schematic and logic diagrams for CD4011B.



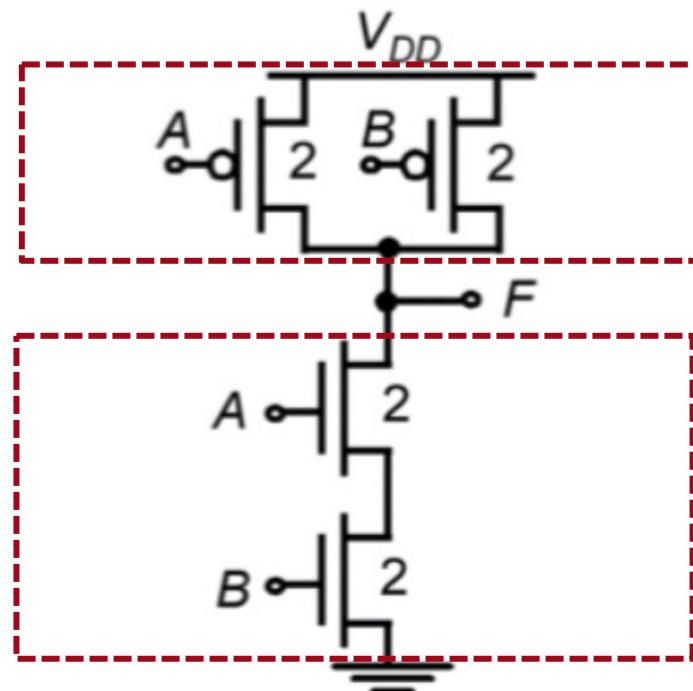
Built in IC: from schematics to layered design



COE COLLEGE[®]

Built in IC: schematics

Logic for NAND: $\overline{A \times B} = \bar{A} + \bar{B}$

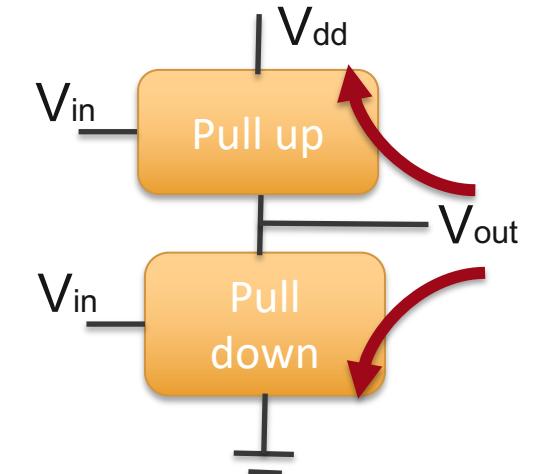


Actual function in the PMOS block

$$\text{ON: } \bar{A} + \bar{B}$$

Actual function in the NMOS block

$$\text{ON: } A \times B$$

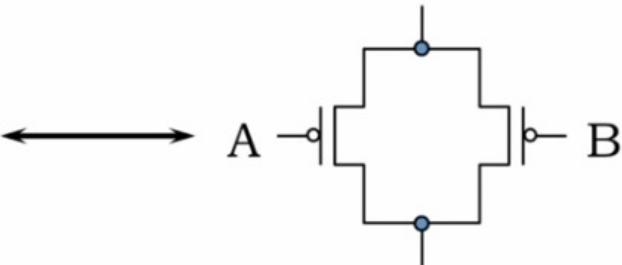
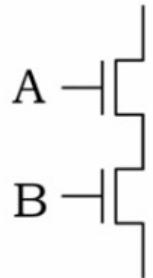


CMOS schematic design is so confined, there's no need for truth table.



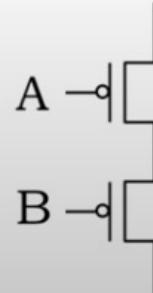
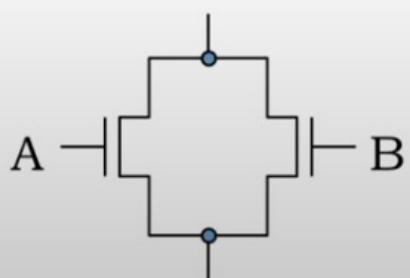
COE COLLEGE[®]

Reminder: C is for complimentary



conducts when A is high
and B is high: $A \cdot B$

conducts when A is low
or B is low: $\bar{A} + \bar{B} = \bar{A} \cdot \bar{B}$

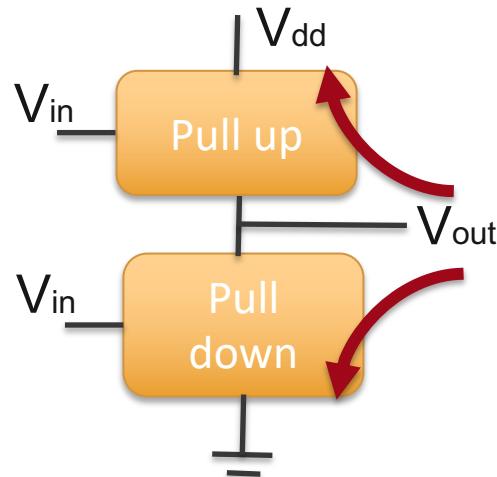


conducts when A is high
or B is high: $A + B$

conducts when A is low
and B is low: $\bar{A} \cdot \bar{B} = \bar{A} + B$

CMOS schematic design, no need for truth table

Target logic operation: $O = \overline{(A+B) \cdot C}$



Step 1 (PMOS block): operate for logic high, but with all \bar{X}

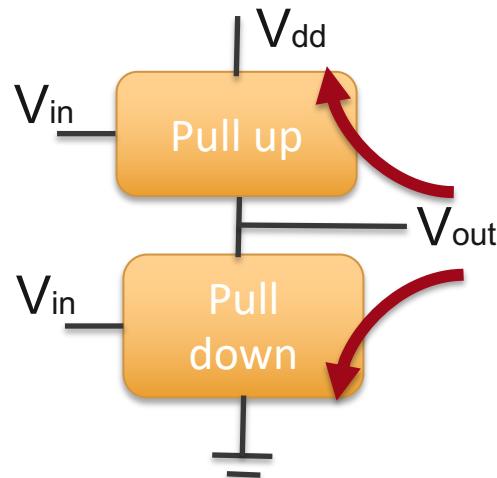
Step 2 (NMOS block): operate for logic low, thus we
are solving for \bar{O}



COE COLLEGE[®]

CMOS schematic design, no need for truth table

Target logic operation: $O = \overline{(A+B) \cdot C}$



Step 1 (PMOS block): operate for logic high, but with all \bar{X}

$$(\bar{A} \cdot \bar{B}) + \bar{C}$$

Step 2 (NMOS block): operate for logic low, thus we
are solving for \bar{O}

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

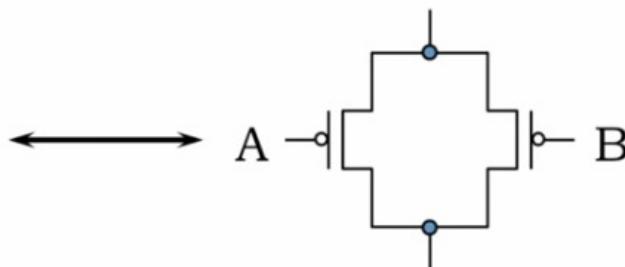
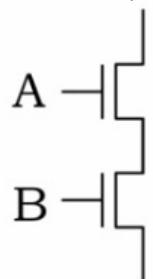


Or quicker way if we remembered the complimentary rules:

The complimentary rule:

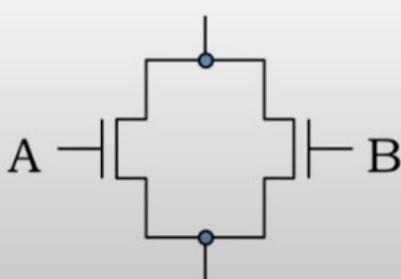
Parallel in NMOS, serial in PMOS

Serial in NMOS, parallel in PMOS

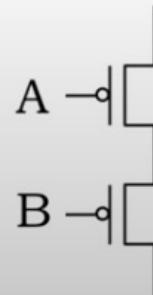


conducts when A is high
and B is high: $A \cdot B$

conducts when \bar{A} is low
or B is low: $\bar{A} + B = A \cdot B$



conducts when A is high
or B is high: $A + B$



conducts when \bar{A} is low
and B is low: $\bar{A} \cdot \bar{B} = A + B$

$$O = \overline{(A+B) \cdot C}$$

Step 1:

Pull down (NMOS)

$$\overline{O} = (A+B) \cdot C$$

A diagram showing the transformation of the expression $(A+B) \cdot C$ into $(\bar{A} \cdot \bar{B}) + \bar{C}$. It consists of two red dashed boxes. The first box encloses the term $(A+B)$, with a red arrow pointing from it to the term $(\bar{A} \cdot \bar{B})$. The second box encloses the term C , with a red arrow pointing from it to the term \bar{C} .

Step 2:

Pull up (PMOS),
just De Morgan it:



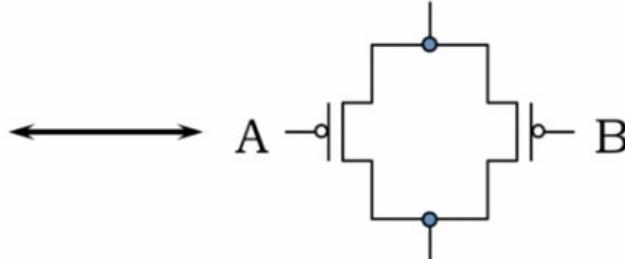
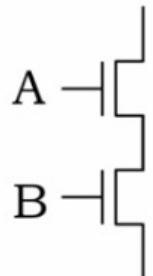
COE COLLEGE®

Just write it down, no truth table

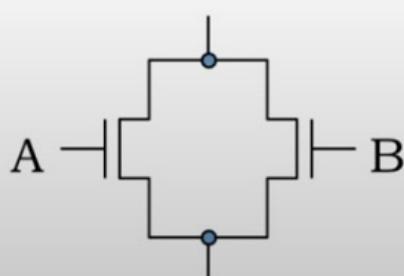
The complimentary rule:

Parallel in NMOS, serial in PMOS

Serial in NMOS, parallel in PMOS

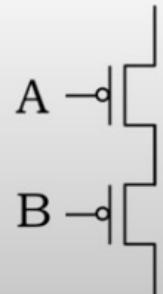


conducts when A is high
and B is high: $A \cdot B$



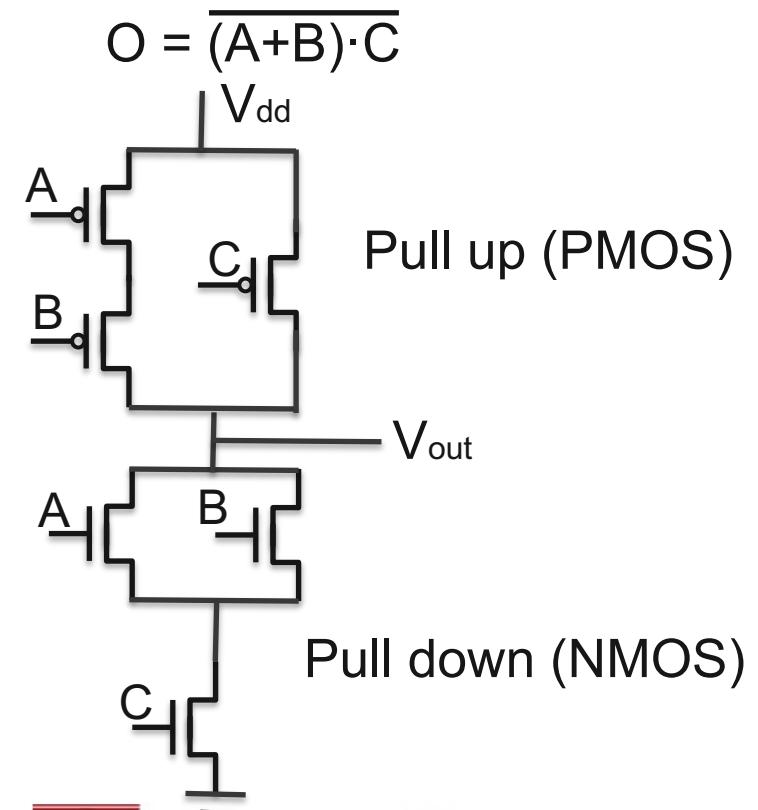
conducts when A is high
or B is high: $A + B$

conducts when \bar{A} is low
or B is low: $\bar{A} + \bar{B} = A \cdot B$



conducts when \bar{A} is low
and B is low: $\bar{A} \cdot \bar{B} = A + B$

CMOS design without truth table



Pull up (PMOS)

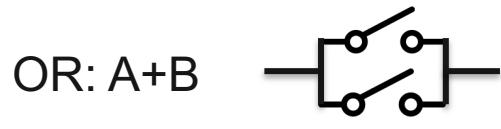
Pull down (NMOS)



COE COLLEGE®

Improved cheat sheet for CMOS design

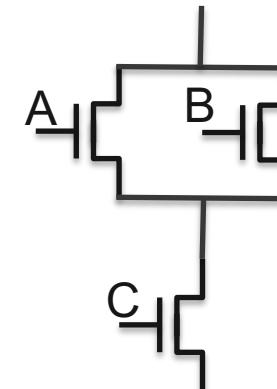
AND OR logic expressed in mechanical Switches:



$$O = \overline{(A+B) \cdot C}$$

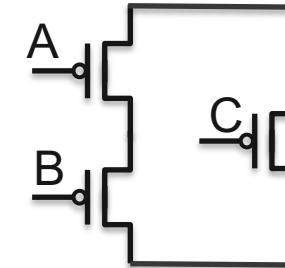
Step 1 NMOS, solve for \overline{O}

$$\overline{O} = (A+B) \cdot C$$



As if they are mechanical switches!

Step 2 PMOS, use the complimentary rule:
Parallel in NMOS, serial in PMOS
Serial in NMOS, parallel in PMOS



Step 3 add everything else (I/O, Vdd, GND and wires)

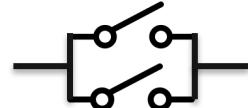


COE COLLEGE[®]

CMOS design, in-class exercise

AND OR logic expressed in
mechanical Switches:

OR: A+B



AND: A·B



$$O = \overline{(A \cdot B + C)} \cdot D + E$$



COE COLLEGE®

Built in IC: to layered design

<https://app.siliwiz.com/>

- [Introduction to SiliWiz](#)

How SiliWiz helps you get an understanding of how semiconductors work and are manufactured

- [Draw a Resistor](#)

Learn the basics of SiliWiz while creating a component

- [Parasitics](#)

How we get unwanted parasitics in circuits and what that means

- [Voltage Divider](#)

Divide a number with 2 resistors

- [Draw a capacitor](#)

Understand how capacitance changes and using DRC to check your design

- [Draw an N MOSFET](#)

Active components: draw a MOSFET and make measurements

- [Making a logic inverter](#)

Create a logic gate and discover its shortcomings

- [Draw a P type MOSFET](#)

Learn about the N type MOSFET's complement, the P type MOSFET

- [Draw a CMOS inverter](#)

Understand the benefit of CMOS and explore standard cells



COE COLLEGE®

Built in IC: to layered design

<https://app.siliwiz.com/>

SILIWIZ

Preset
inverter.json

LOAD SAVE CLEAR STL

Layers

active

- p substrate
- n well
- n diffusion
- p diffusion
- p tap
- n tap

passive

- polysilicon
- polyres
- metal1
- mim capacitor
- metal2

via

- metal1 via
- metal2 via

0um 4.5um 9um

CROSS SECTION & DRC ✓ SIMULATION

Cross Section View

DRC Errors

✓ DRC OK

CROSS SECTION & DRC ✓ SIMULATION

Plot signals: in (x) out (x) +

Input voltage:

- Min: 0V
- Max: 5V

Pulse delay: 0ps

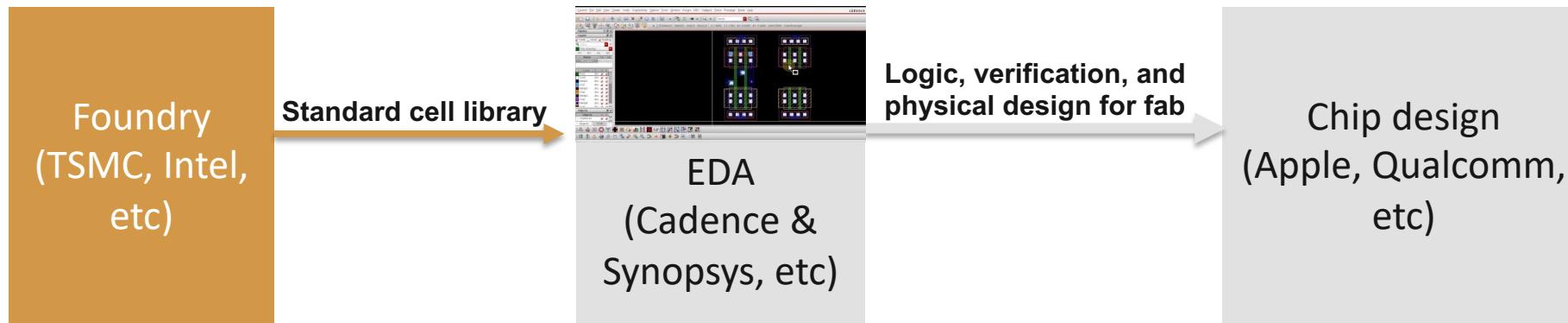
Rise time: 50μs

Time scale: 60μs

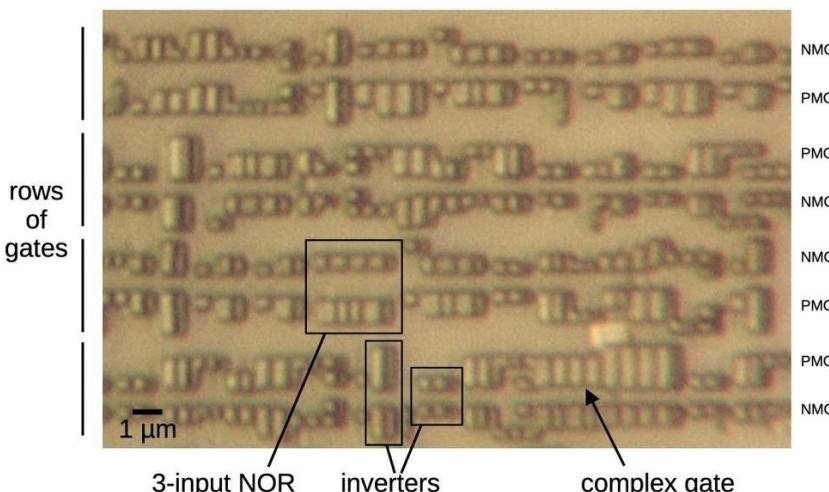


COE COLLEGE®

CMOS design for large circuitry



Standard cell logic form a 180 nm CMOS chip in the 90s



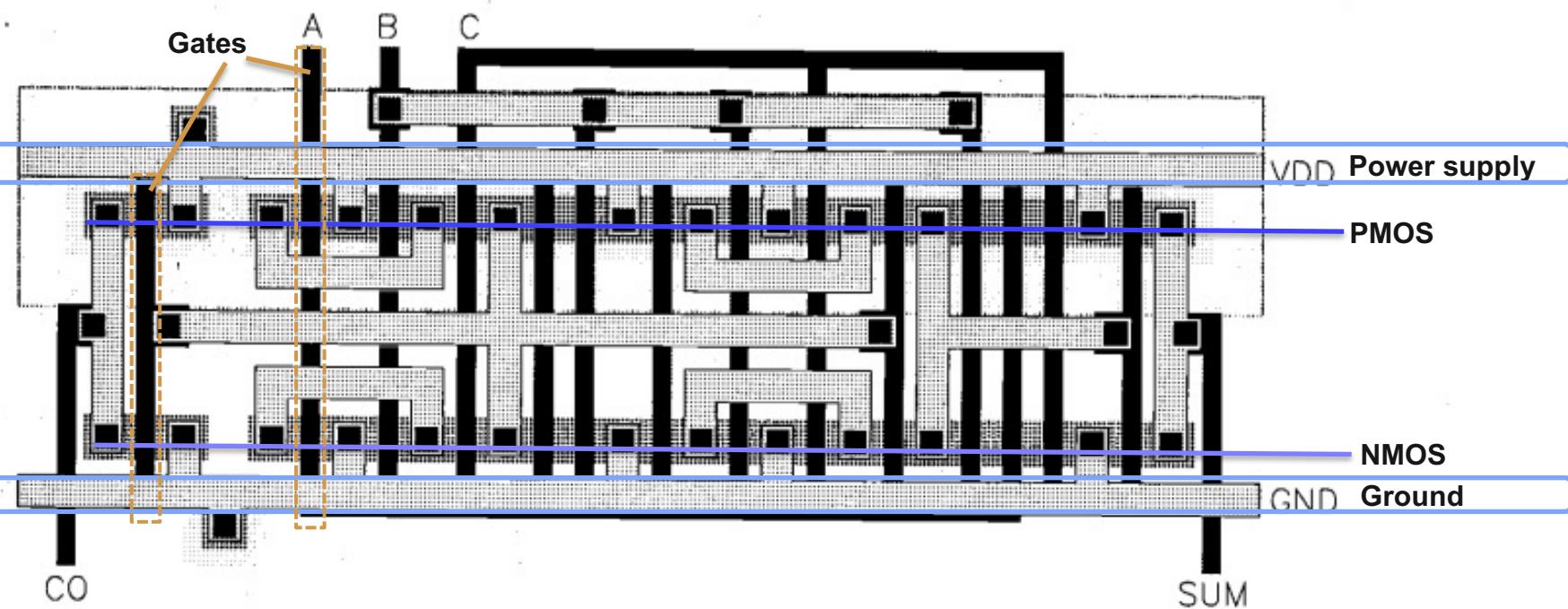
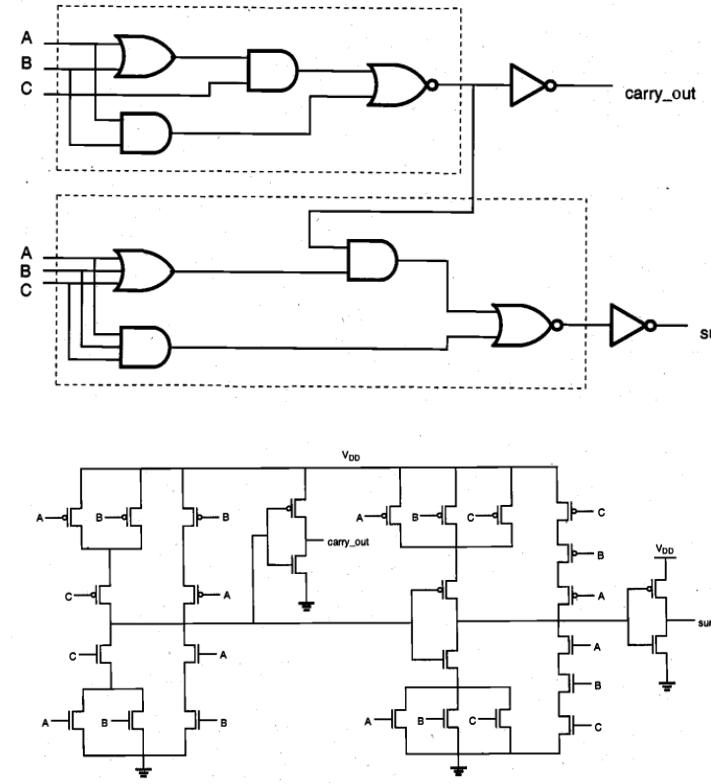
<https://www.righto.com/2024/06/montreal-mifare-ultralight-nfc.html>



COE COLLEGE[®]

CMOS design rules shown as a full-adder

or being easily adaptable to computer-aided design (CAD). The silicon area occupied by this full-adder layout is $(21 \mu\text{m} \times 54 \mu\text{m}) = 1134 \mu\text{m}^2$.



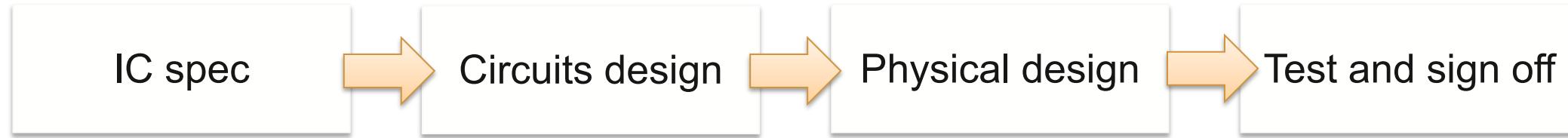
Mano, M. Morris. *Digital logic and computer design*. 1979

E_®

Figure 1.10. Initial layout of the full-adder circuit using minimum-size transistors.

Design our own IC, yes we can!

The general process flow

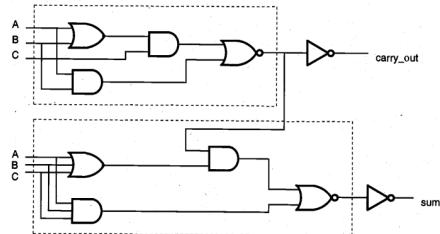


IC spec

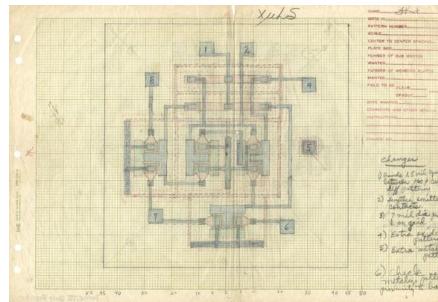
- High level requirements
- Customer needs

Circuits design

- Logic circuits
- Memory circuits
- Netlist of every groupings

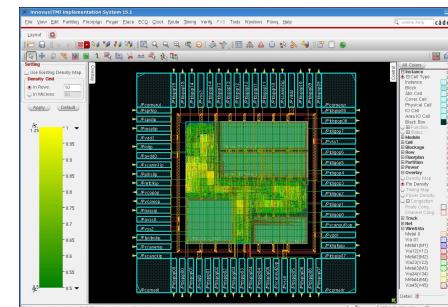


Physical design



Test and sign off

Hand draw mask layout, Fairchild 1960
<https://www.computerhistory.org/revolution/digital-logic/12/287>



cadence



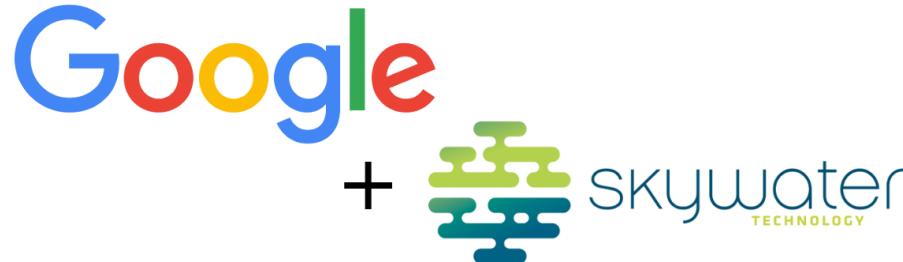
COE COLLEGE®

Design our own IC, yes we can!

<https://tinytapeout.com/siliwiz/>

<https://app.siliwiz.com/>

<https://github.com/ncsu-eda/FreePDK3>



FOSS 130nm Production PDK
github.com/google/skywater-pdk

FreePDK3

Design Rule Manual

Revision: 1.0

Process Node: 3nm

Release Date: August 30, 2021

Copyright © 2021 Sushant Sadangi and W. Rhett Davis, North Carolina State University

This manual is licensed under Creative Commons Attribution 4.0 (CC BY 4.0) license.

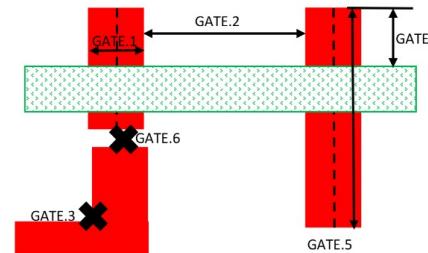


- PDK are design rules set by the foundry. With opensource support, it is technically possible to generate a design file (GDS) and get your IC chips fabricated.

Example of design rules from FreePDK3 by NCSU, 2021

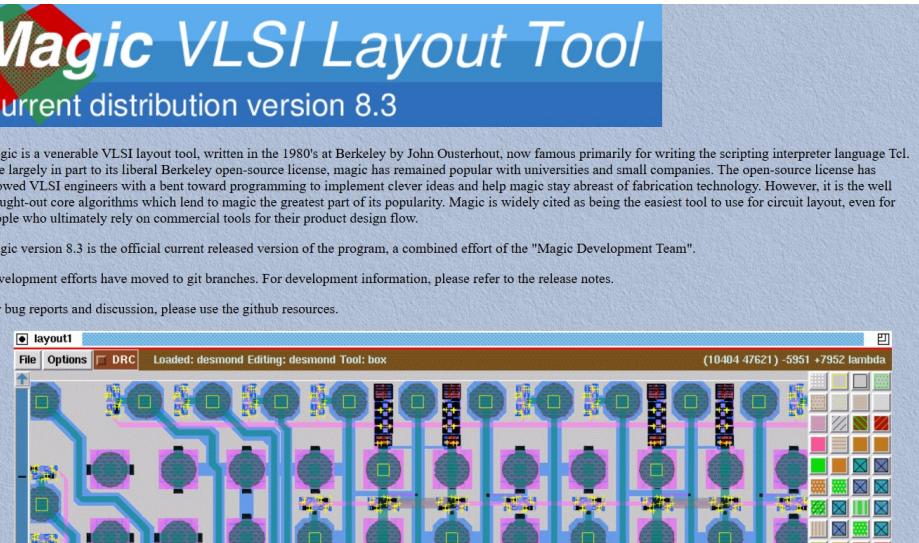
2.6 Gate Layer

Rule	Value	Description
GATE.1	15 nm	GATE exact horizontal width
GATE.2	27 nm	Minimum horizontal spacing between GATE or DUMMY layers
GATE.3		GATE may not bend
GATE.4	21.5 nm	GATE min extension past ACT
GATE.5	40 nm	GATE minimum vertical length
GATE.6		GATE may not be discontinuous along the vertical axis. Use GCUT layer to mark cuts in the GATE
GATE.7		ACT layer vertical edge may not lie inside, or coincide with, the GATE layer
GATE.8	6 nm	Minimum horizontal spacing between ACT and GATE (not cut by GCUT and not interacting with ACT)



COE COLLEGE®

Into the realm of digital design



Magic VLSI Layout Tool
Current distribution version 8.3

Magic is a venerable VLSI layout tool, written in the 1980's at Berkeley by John Ousterhout, now famous primarily for writing the scripting interpreter language Tcl. Due largely in part to its liberal Berkeley open-source license, magic has remained popular with universities and small companies. The open-source license has allowed VLSI engineers with a bent toward programming to implement clever ideas and help magic stay abreast of fabrication technology. However, it is the well thought-out core algorithms which lead to magic the greatest part of its popularity. Magic is widely cited as being the easiest tool to use for circuit layout, even for people who ultimately rely on commercial tools for their product design flow.

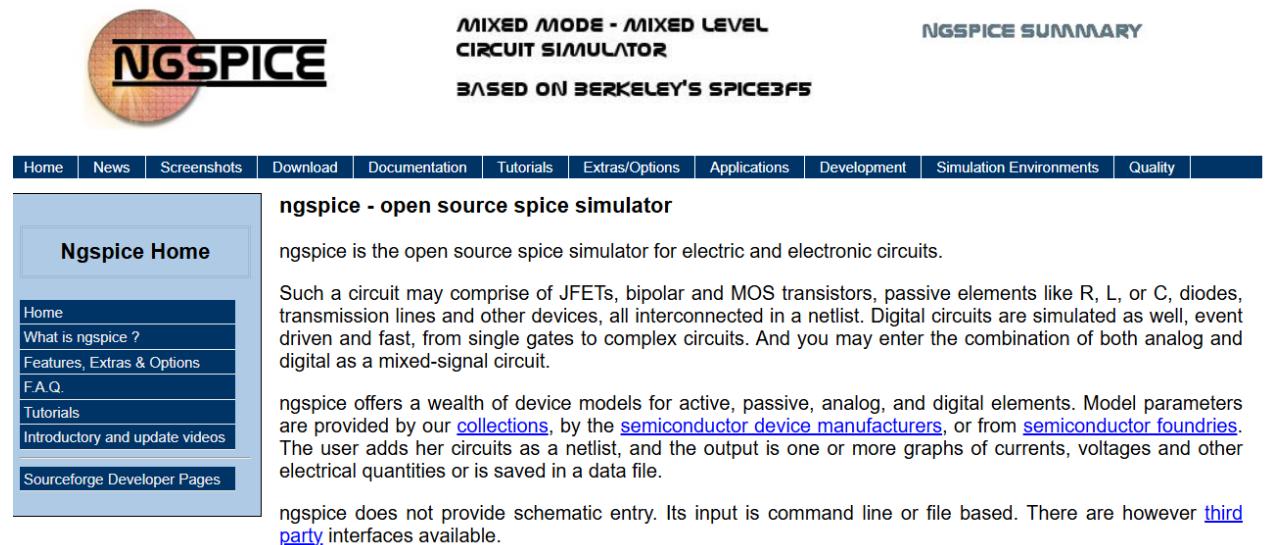
Magic version 8.3 is the official current released version of the program, a combined effort of the "Magic Development Team".

Development efforts have moved to git branches. For development information, please refer to the release notes.

For bug reports and discussion, please use the github resources.

layout1
File Options DRC Loaded: desmond Editing: desmond Tool: box (10404 47621) -5951 +7952 lambda.

Welcome
Download
Install
Release Notes
Code History
Using Magic
Technology Files
Documentation
Development
GitHub
OCD Home
Non-frames version



NGSPICE
MIXED MODE - MIXED LEVEL CIRCUIT SIMULATOR
BASED ON BERKELEY'S SPICE3F5

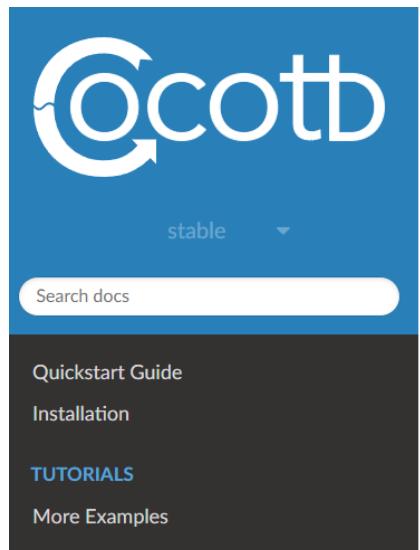
Home News Screenshots Download Documentation Tutorials Extras/Options Applications Development Simulation Environments Quality

ngspice - open source spice simulator

ngspice is the open source spice simulator for electric and electronic circuits. Such a circuit may comprise of JFETs, bipolar and MOS transistors, passive elements like R, L, or C, diodes, transmission lines and other devices, all interconnected in a netlist. Digital circuits are simulated as well, event driven and fast, from single gates to complex circuits. And you may enter the combination of both analog and digital as a mixed-signal circuit.

ngspice offers a wealth of device models for active, passive, analog, and digital elements. Model parameters are provided by our [collections](#), by the [semiconductor device manufacturers](#), or from [semiconductor foundries](#). The user adds her circuits as a netlist, and the output is one or more graphs of currents, voltages and other electrical quantities or is saved in a data file.

ngspice does not provide schematic entry. Its input is command line or file based. There are however [third party](#) interfaces available.



cocotb
stable ▾
Search docs

Quickstart Guide
Installation
TUTORIALS
More Examples

Welcome to cocotb's documentation!

[View page source](#)

Welcome to cocotb's documentation!

What is cocotb?

cocotb is a COroutine based COsimulation TestBench environment for verifying VHDL and SystemVerilog RTL using Python.

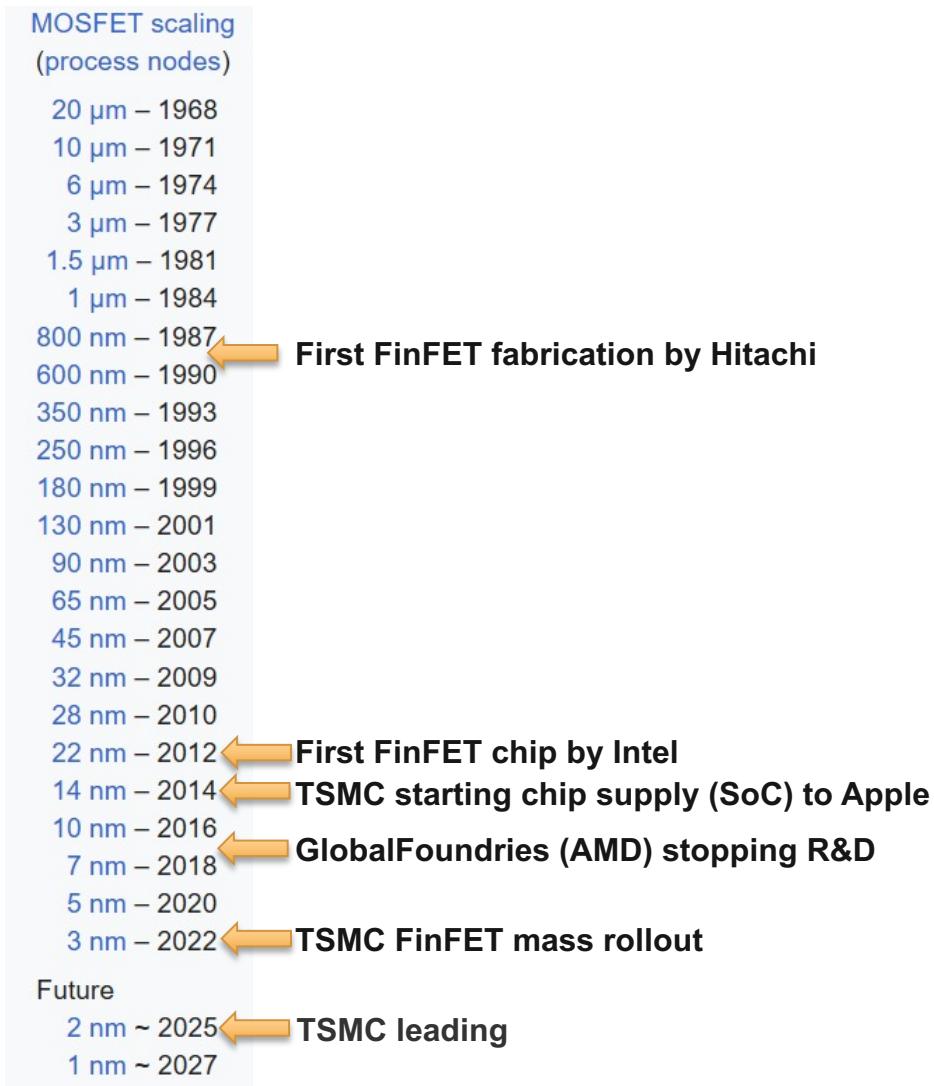
cocotb is completely free, open source (under the [BSD License](#)) and hosted on [GitHub](#).

cocotb requires a simulator to simulate the [HDL](#) design and has been used with a variety of simulators on Linux, Windows and macOS. Please check the [Simulator Support](#) page for specifics.



COE COLLEGE®

The state of the art for CMOS: The “nm” race

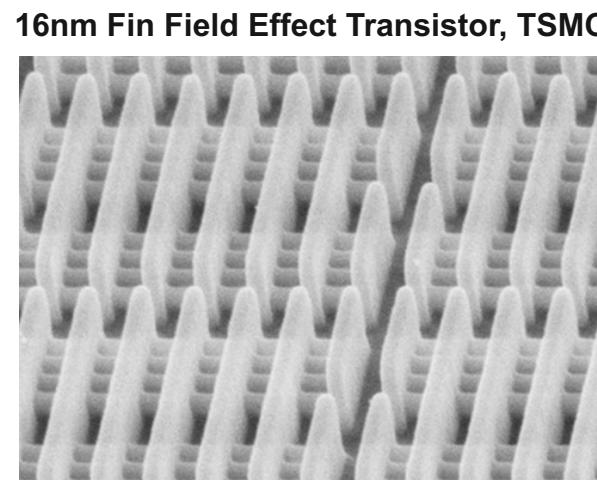
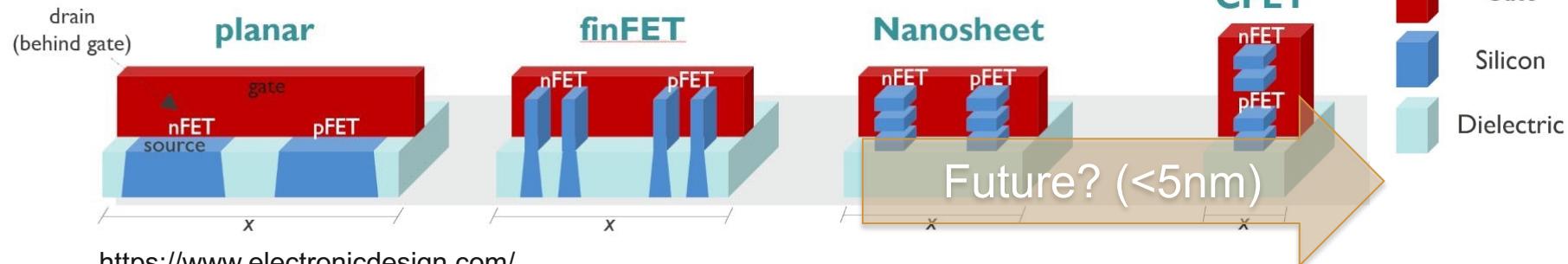


COE COLLEGE®

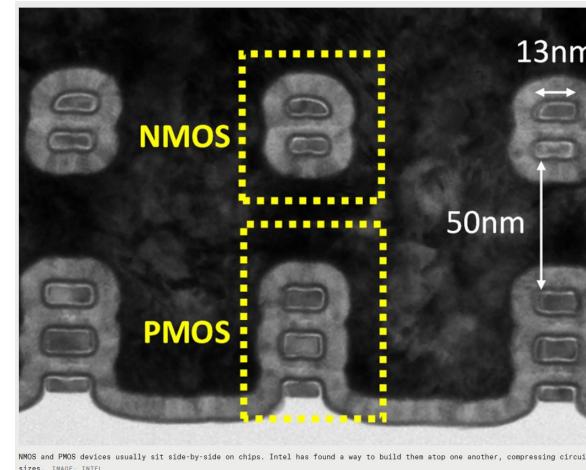
The state of the art for CMOS:

The “nm” race

MOSFET scaling (process nodes)	
20 μm	– 1968
10 μm	– 1971
6 μm	– 1974
3 μm	– 1977
1.5 μm	– 1981
1 μm	– 1984
800 nm	– 1987
600 nm	– 1990
350 nm	– 1993
250 nm	– 1996
180 nm	– 1999
130 nm	– 2001
90 nm	– 2003
65 nm	– 2005
45 nm	– 2007
32 nm	– 2009
28 nm	– 2010
22 nm	– 2012
14 nm	– 2014
10 nm	– 2016
7 nm	– 2018
5 nm	– 2020
3 nm	– 2022
Future	
2 nm	~ 2025
1 nm	~ 2027



16nm Fin Field Effect Transistor, TSMC

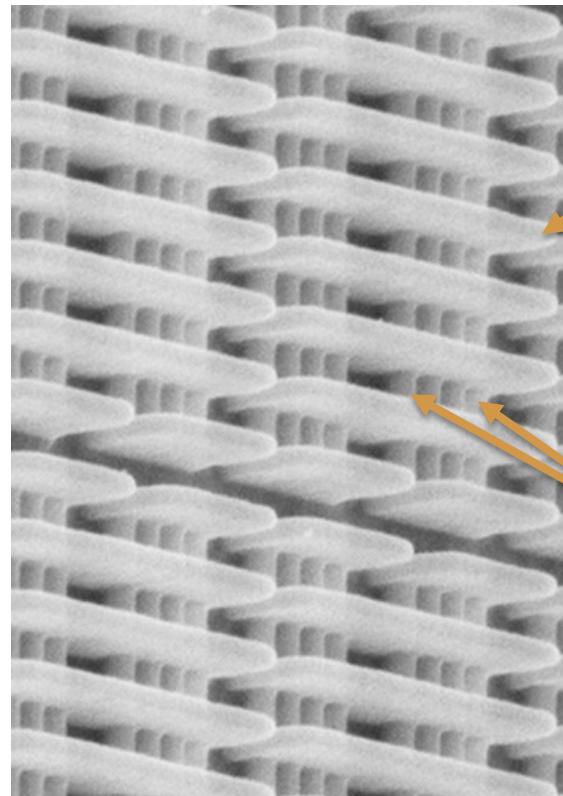


Nanosheet inverter, Intel reported in 2020

<https://spectrum.ieee.org/>



COE COLLEGE®



FinFET

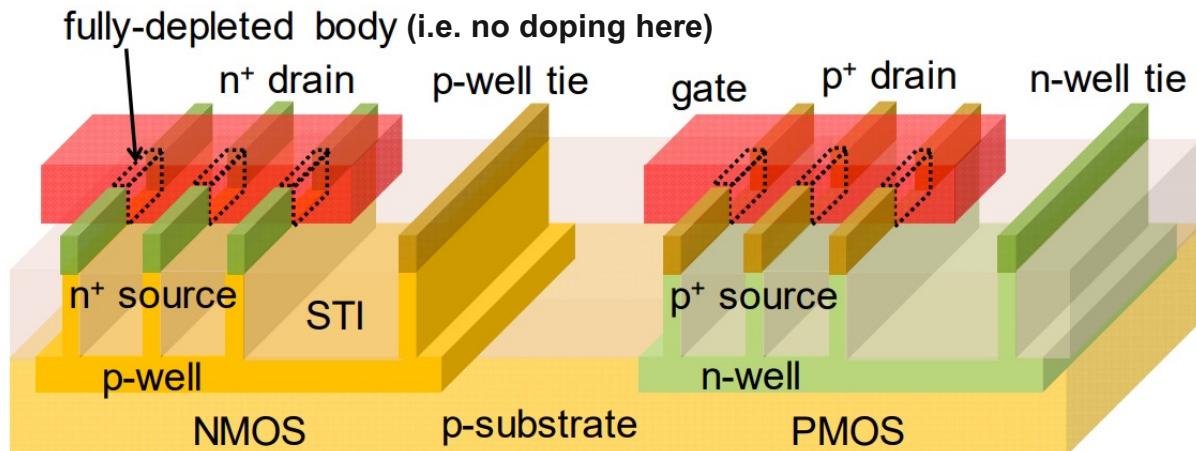


Fig. 1. Fully depleted bulk CMOS finFETs. DOI:[10.1109/CICC.2018.8357060](https://doi.org/10.1109/CICC.2018.8357060)

- The dimension of the fin is confined by the foundry (TSMC).
- The number of the fin can be adjusted to get different gate current.
- The fin stacks and gate can then be cut or connected.

CMOS inverter layout example

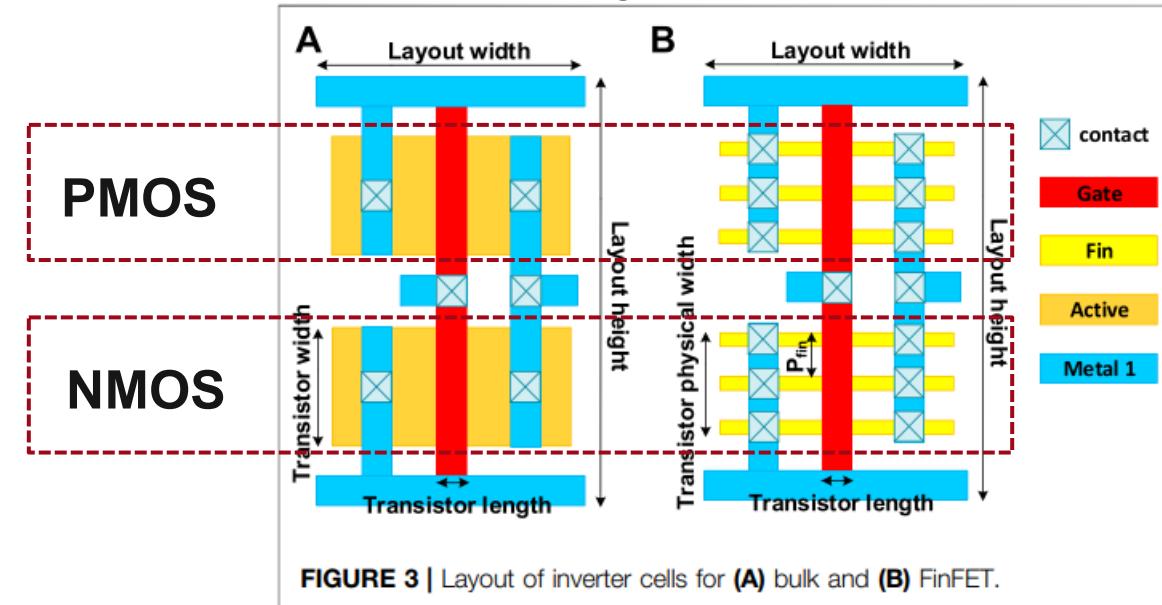
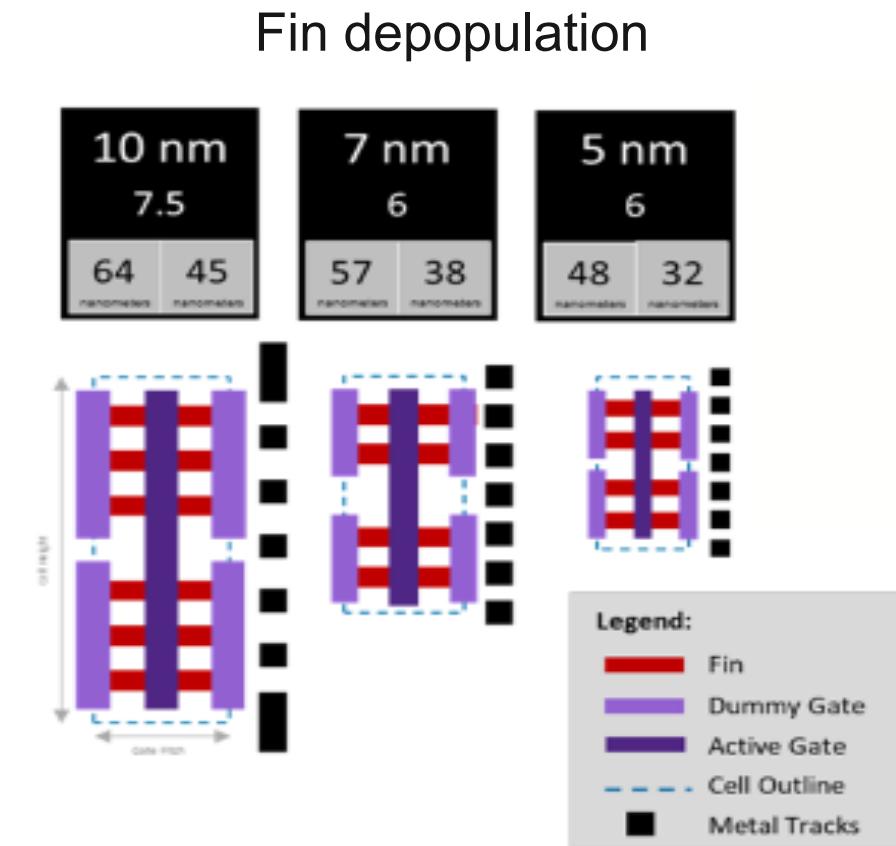
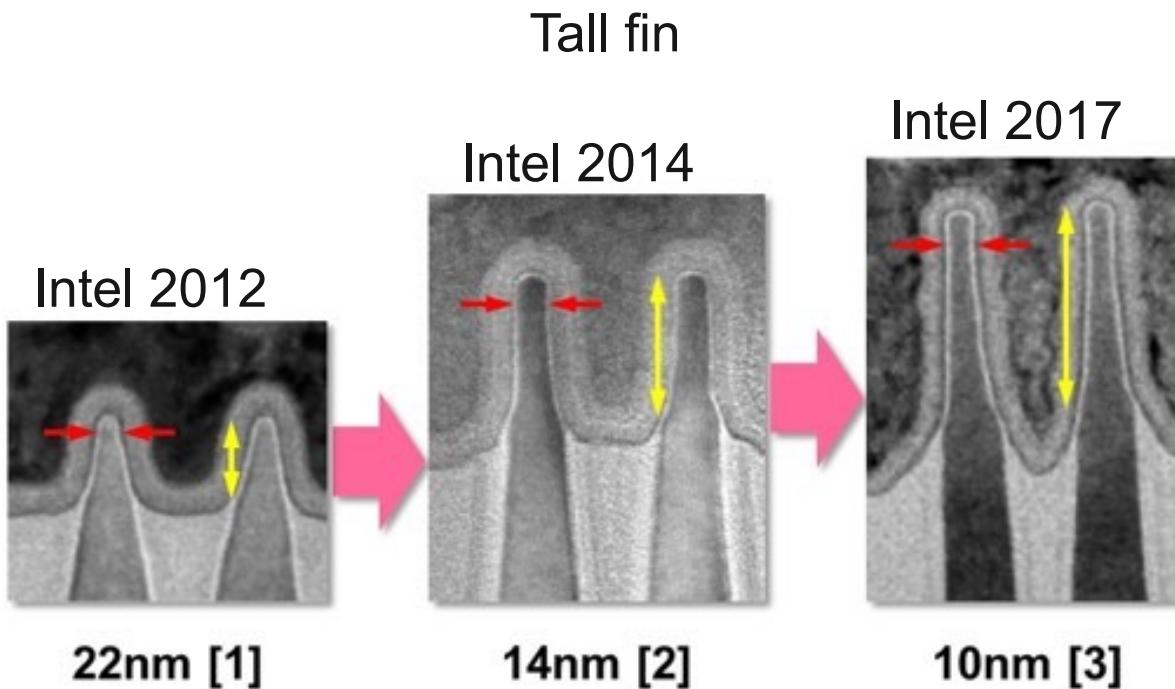


FIGURE 3 | Layout of inverter cells for (A) bulk and (B) FinFET.

FinFET iteration



<https://semiengineering.com/selective-removal-for-stronger-fins/>

<https://www.eetimes.com/what-to-expect-at-5-nm-and-beyond-and-what-that-means-for-eda/>



COE COLLEGE[®]