FABS, DESIGN RULES AND THE ART OF LAYOUT

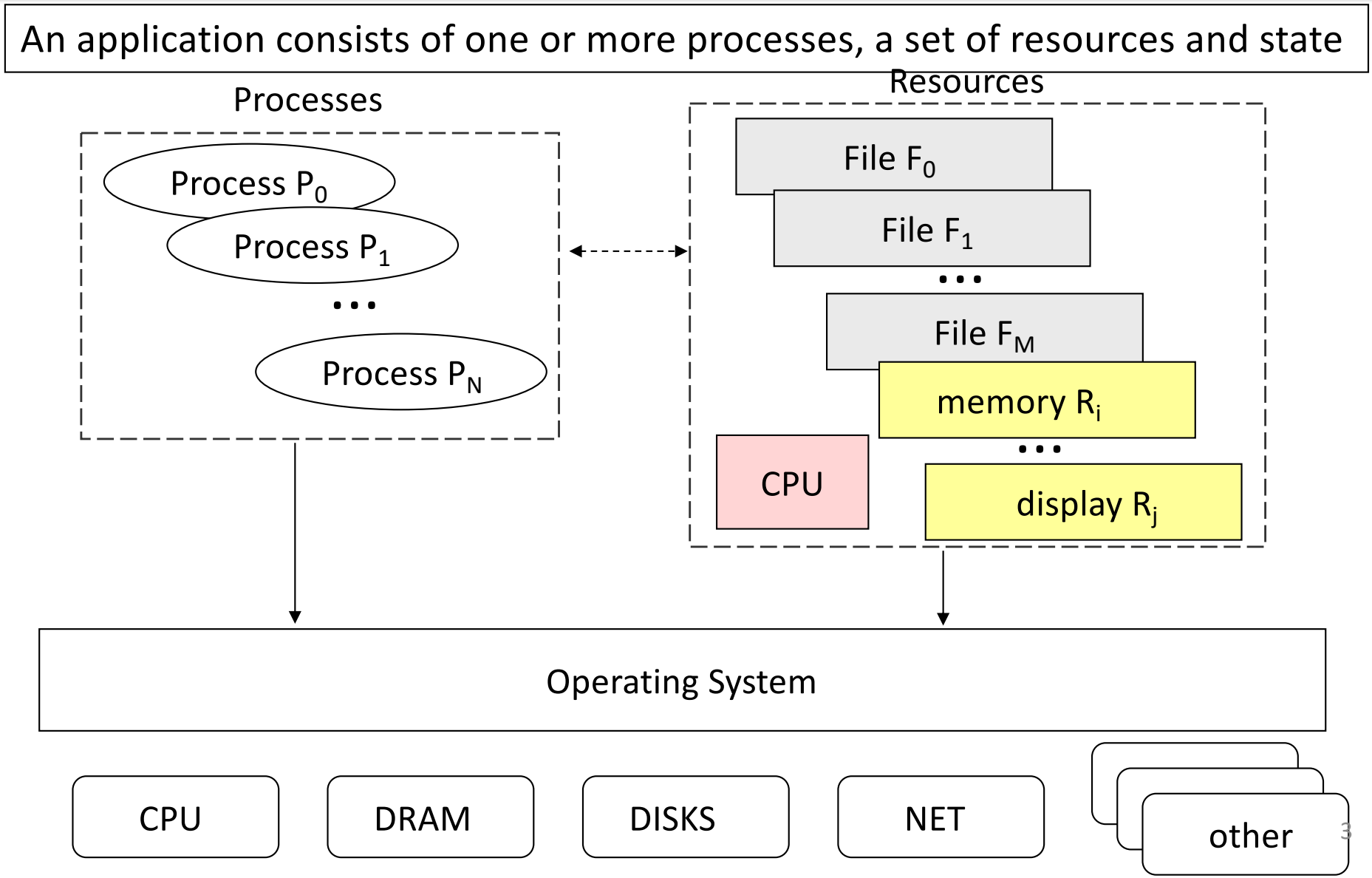
James E. Stine, Jr.
Edward Joullian Endowed Chair in Engineering
Oklahoma State University
Electrical and Computer Engineering Department
Stillwater, OK 74078 USA
james.stine@okstate.edu

Tools and their Use!

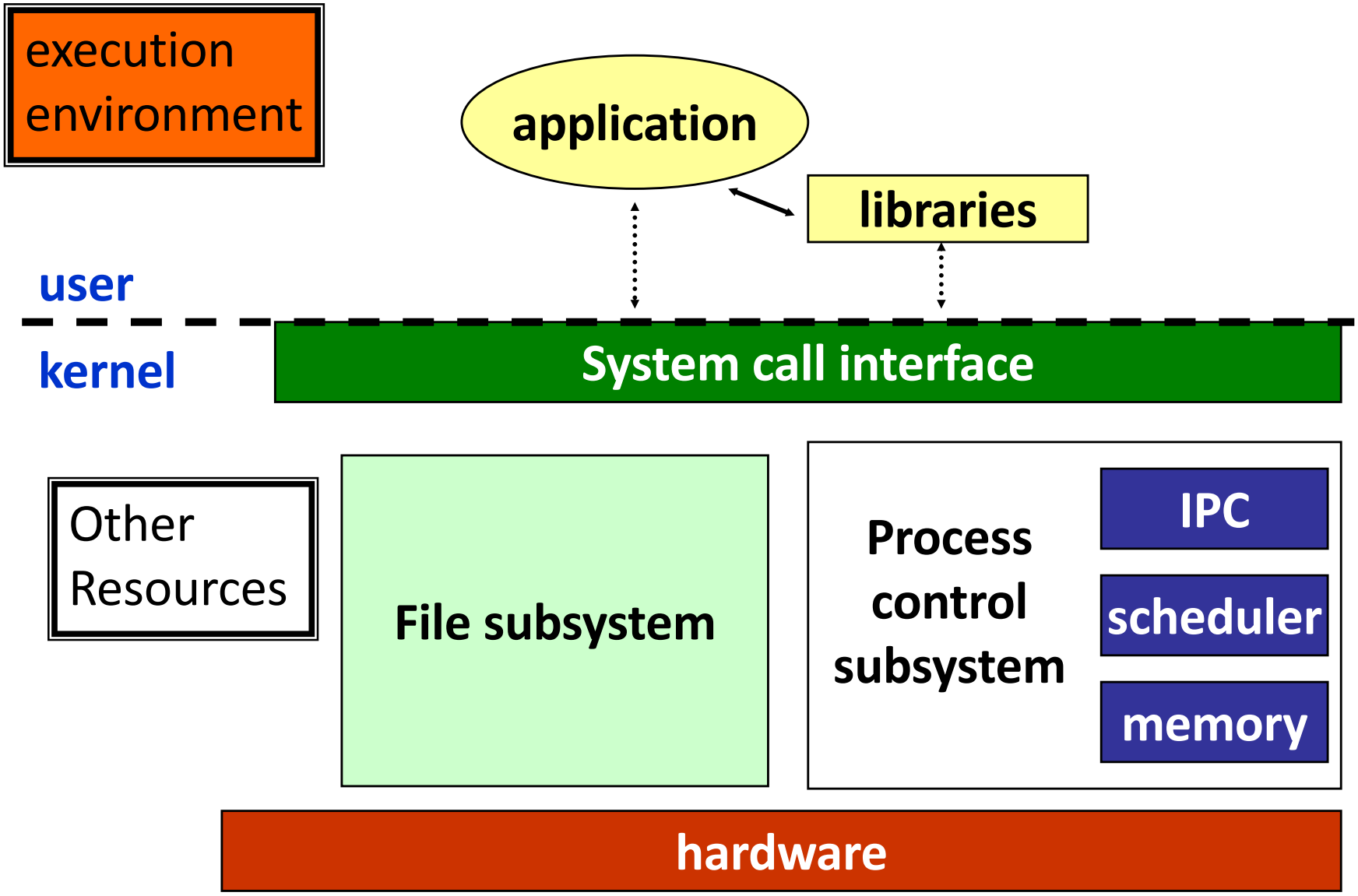
- Scripts?
  - What is a script and why should I use them?
  - A script is a set of instructions that enables a user to repeat a set of actions, usually with a keystroke, program, or button.
- Program, Process and how to get elements to interact?
  - The environment!!!
  - Let's take a look-see before we begin!!!!

we need the port number! using vnc

Processes and Resources



Traditional OS (UNIX) Model



Shell Overview

What is a Shell?

- UNIX shells provide a "command line" interface which allows the user to enter commands which are translated by the shell into something the kernel can comprehend and then is sent off to the kernel for it to act upon.
- The user can pick their shell (just like the applications, desktop manger, window manager, etc. on a UNIX system).

OSU Shells

- We have decided to give everyone the Turbo C shell, because it's the most popular for our work in this class and what is used in industry.
- Shells available on GL include:
  - tcsh - Turbo C Shell
  - csh - C Shell
  - ksh - Korn Shell
  - bash - Bourne Again Shell
  - sh - Shell

Environment Variables

- Think of the shell as any other program that you write. Your program maintains information about its current state. Since the shell's main job is to act as a liaison between the kernel and the user, it maintains information about the computing environment. The environment variables hold this information.
- All Operating Systems have environmental variables!!!
- Most UNIX systems provide a command **setenv** that will allow you to see all of these variables that the shell is maintaining.
  - Windows : type “set” in a command prompt.
  - All things work for a reason – learn them, love them, enjoy them!

edu-tools!

Example Environment Variables

- **HOME** - your home directory.
- **USER** and **LOGNAME** - your login ID.
- **HOSTNAME** - the name of the host computer.
- **PWD** - the current working directory.
- **LD_LIBRARY_PATH** – location of your (dynamic) libraries.
- **PATH** - a list of directories in which to look for executable commands.

Introducing `tcsh`

- Currently, the default shell on OSU's system is tcsh.
- **tcsh** is short for Turbo C SHell.
- The **.cshrc** file – sometimes **.tcshrc**.
  - csh looks for a configuration file at startup time called "~/.cshrc"
  - **I recommend backing up a copy of your account configuration files before modifying them** (such as "cp .cshrc .cshrc.bak").
  - Changes to your configuration file do not affect the system immediately after you save the file.
    - Any time you open a terminal it invokes your .cshrc file, therefore, any change could easily be invoked by opening a new terminal.
    - Right mouse button → terminal
    - Why do we want to want to use the configuration files?
- I currently use an **alias** to help me with my scripts!
  - `alias rm    "rm –i"`

Schematic, yes, Layout, no

- Okay…we can build any gate using CMOS pairs
- Remember that gates are complementary and inverting!
- Gates are classified usually according to their logic:
  - INV
  - NAND
  - NOR
  - AND
  - OR
  - OAI (Or-And-Invert)
  - AOI (And-Or-Invert)
- Also, sometimes the number of inputs is included in naming convention
  - AOI221
  - NAND2

Typical Processes

- A process is a sequence of steps required to engineer a chip.
- Some important CMOS processes are p-well and n-well.
- Ours is SKY130 or 0.130um @ L~130nm
- Skywater Technology is our fabrication process drawn @ 0.15
- The drawn feature size may or may not be the value in the process definition because it is a *fabrication* process.
- All transistors will have this minimal length to guarantee its operability.

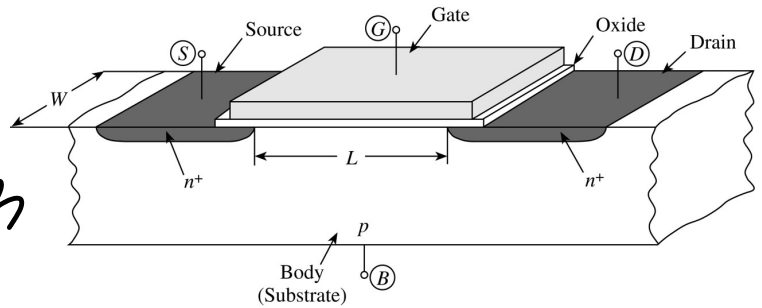


FIGURE 1.30 Simplified structure of an n-channel MOS transistor.

PDK

skywater

https://www.skywatertechnology.com

Process Design Kit!

Design Rules

Who needs them?

- There are practical limits on the resolution to which we can manufacture Silicon chips.
- These have to do with equipment precision, mask alignment, lateral diffusion, etc.
- To achieve reasonable yield, a layout must not violate these limits.
- To shield the IC designer from the complexities of the fabrication process, process engineers specify simple design rules

to have the highest yield

Lynn Conway

- Before 1999, Lynn Conway was already well respected for her many accomplishments:
  - VLSI work at Xerox PARC
  - DARPA / Strategic Defense Initiative
- Her work at IBM included the invention of a fundamental component of today's modern superscalar computers.

45 Years since VLSI was first taught!

Scalable! VLSI

CMOS

SCMOS

Lambda

$\lambda = \frac{1}{2}$ allows to change in the process but keep overall design the same!

General Layout Layers

- Many more varieties of processes are possible.
- New processes are constantly being developed.
- Some terminology:
  - Pitch: dimension in the metal to describe interdimensions. Pitch is used to mean the cell dimension perpendicular to the stacking direction.
  - Floorplan: high level organization of design blocks.
  - Ripper: A connector from a line to a bus.
  - Instance: the actually symbol instance.
- Design Rules: A set of permissible geometries and geometric constraints (a recipe)
- Include minimum width, minimum separations, and minimum overlaps.
- Usually, basic unit of length is λ .
- For a 2 µm process, λ = 1 µ m

Layout Design Rules

- Layout design rules describe device dimensions and how close different and like layers can be brought to close proximity without creating shorts.
- The lambda-based design rules made popular by Mead and Conway are based on the single parameter λ and permit for ease of scaling.
- Industry used micron as a unit of measure, and this makes scaling difficult since not all dimensions do not scale uniformly.

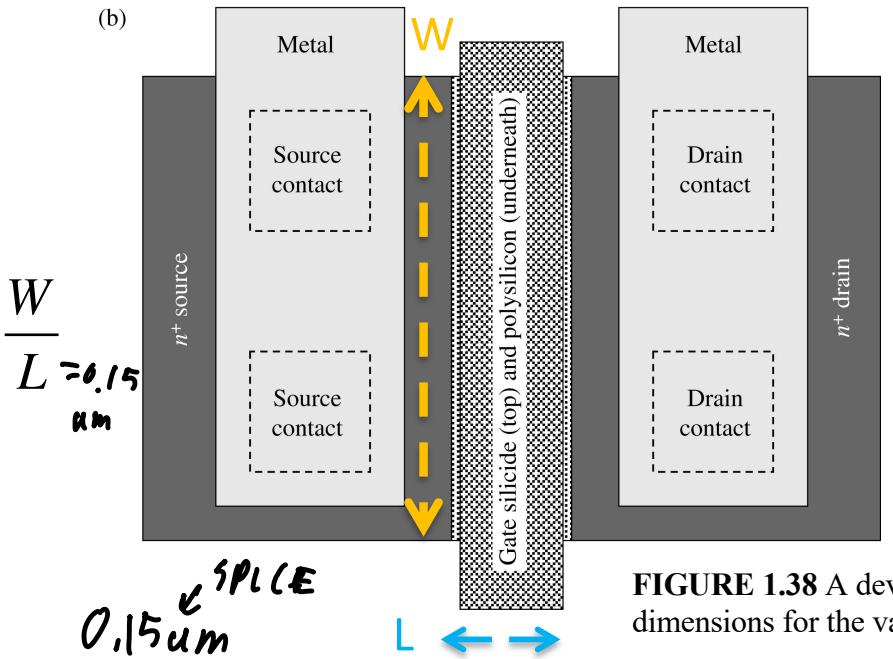
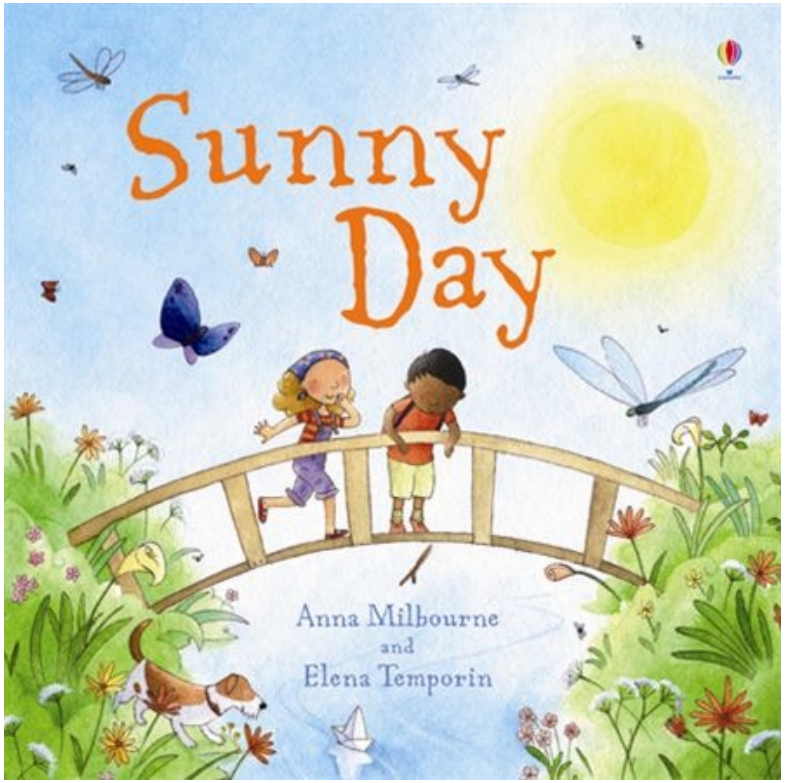


FIGURE 1.38 A device drawing showing realistic relative dimensions for the various features; (b) top view.

- Lambda (λ) is generally half of the minimum drawn transistor channel length.
- The channel length describes the distance between the inside edges of the source and drain (what's defined by process/technology).
- It is set by the minimum poly width (what engineers' control).
- We are beginning to refer to the gate channel length in nanometers (nm) whereas for technologies 0.18 μm and above we used the micron.

Art?

- What is it?
- What are we doing?
- What are you doing?



[Usborne]

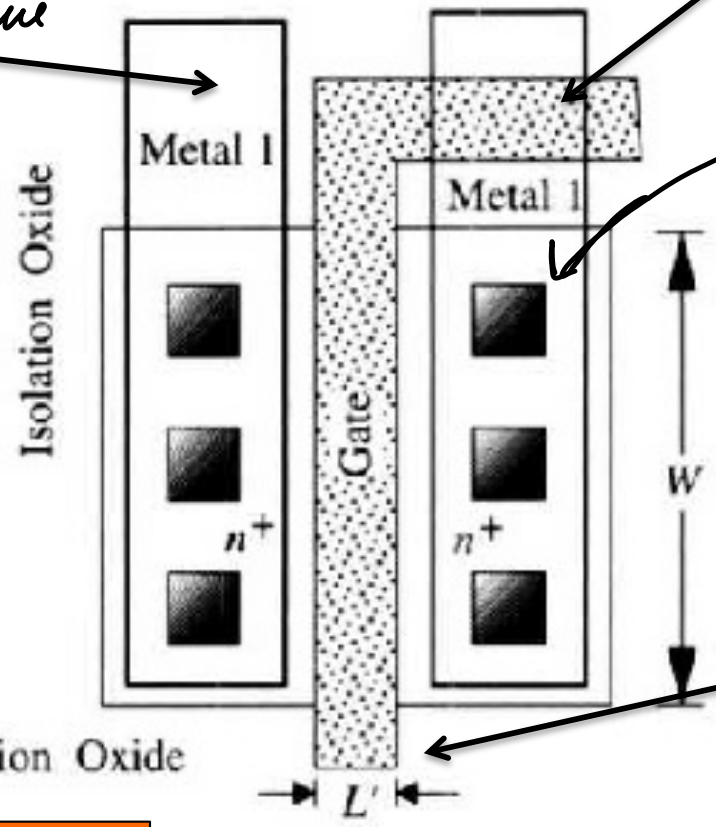
Basic Principle

- The basic principle is the polysilicon across the diffusion area.
- This forms a transistor for every crossing path!

Typically drawn as "blue"

typically drawn as "red"

How many gates?

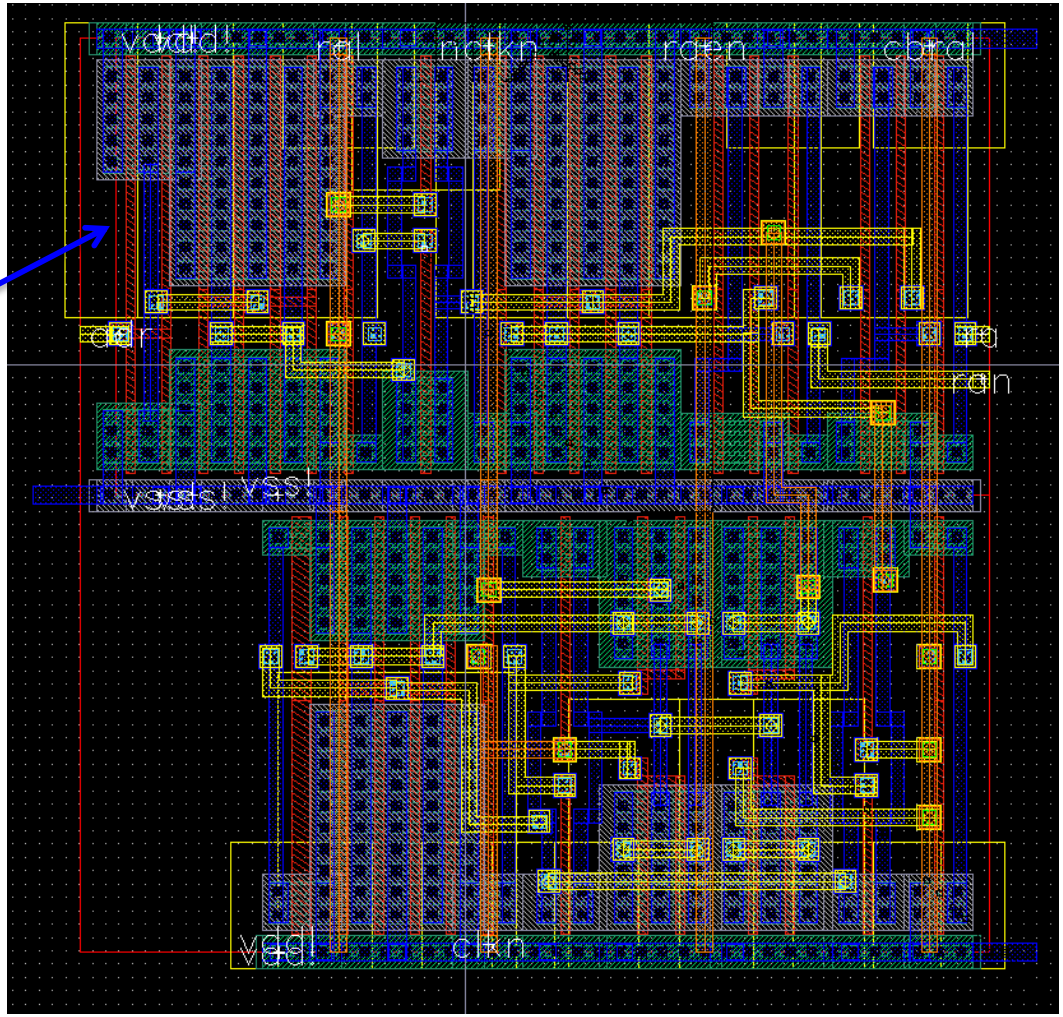


Overhang???????

metal1 is the first layer of metal!

[Uyemora]

How many?



[http://www.eecg.toronto.edu/~roman/teaching/1388/2004/finalProj/2004_ECE1388_FP_www/256kb_SDRAM/]

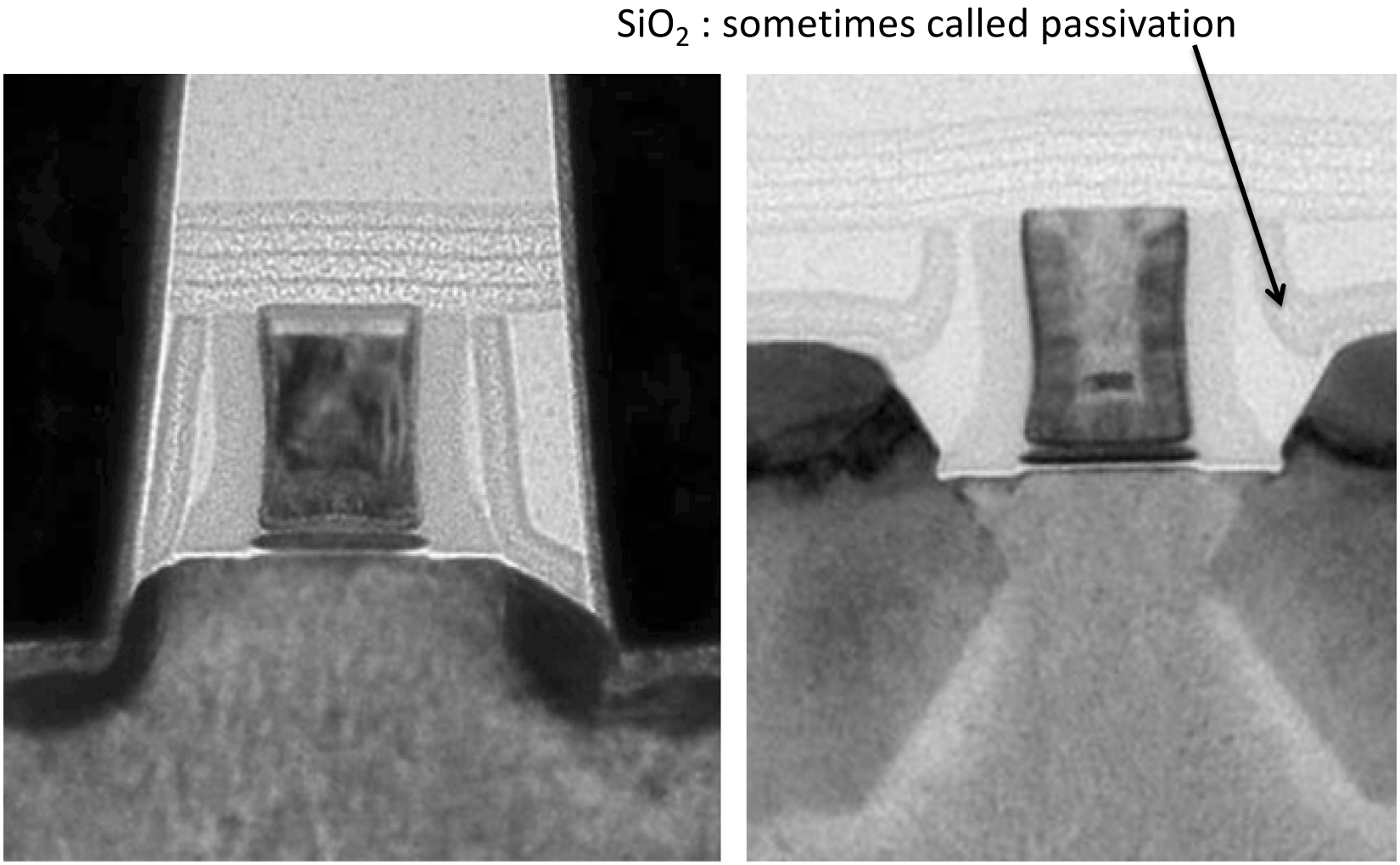
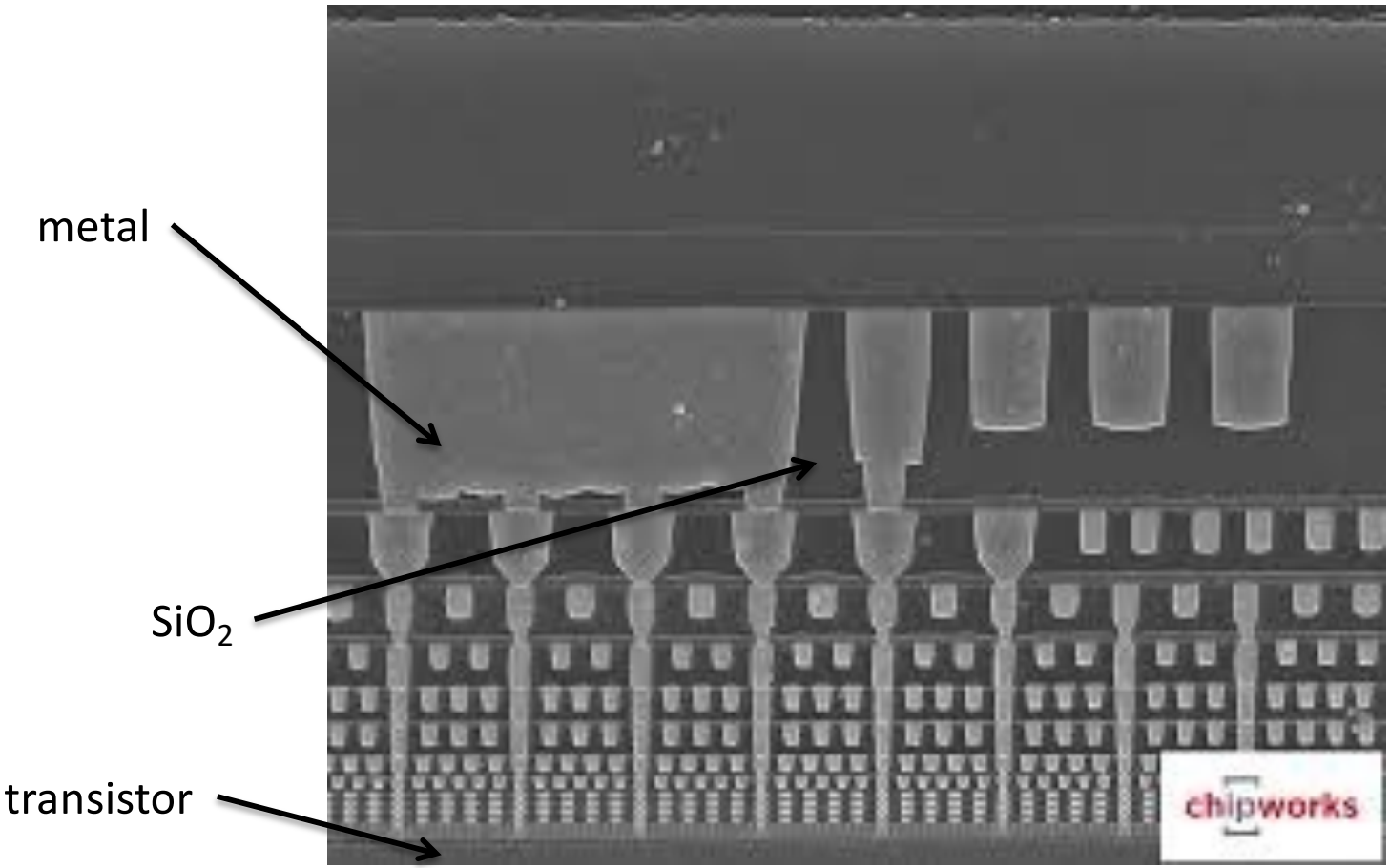


FIGURE 1.40 SEM images of (a) an nMOS and (b) a pMOS transistor in Intel's 45-nm technology. Copyright © 2008 by IEEE, Courtesy of Intel Corporation.

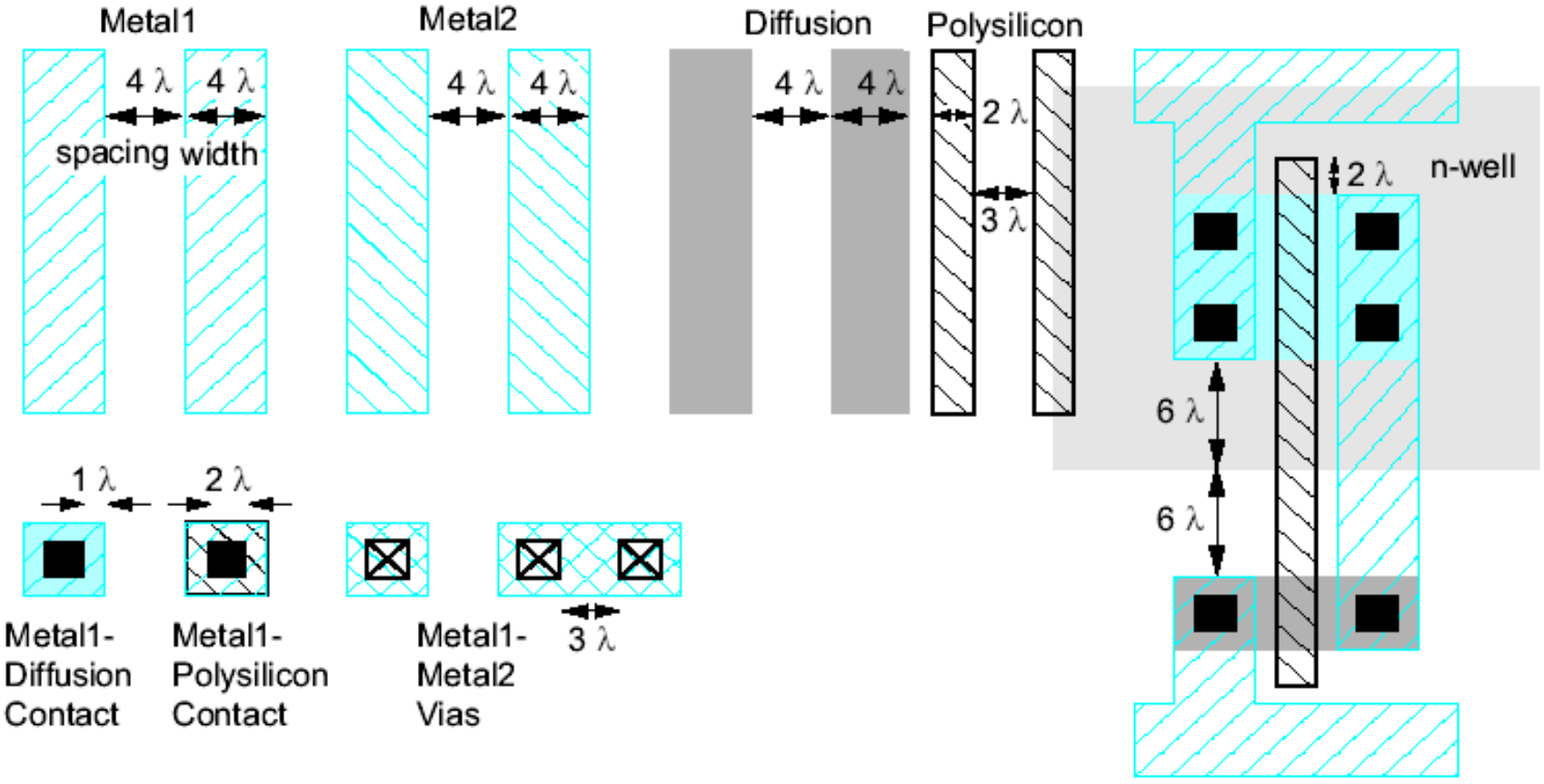
Interconnect (Metal Layers)



[semimd.com]

Simplified Design Rules

- Conservative rules to get you started



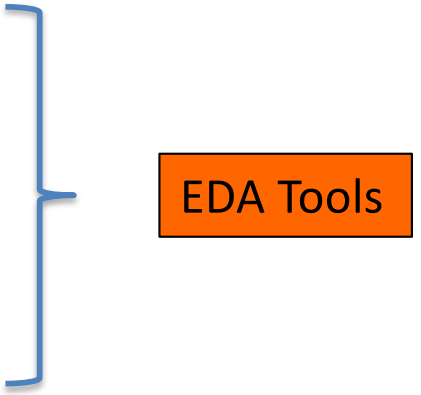
We utilize Lambda (λ) rules in this class – Intel and AMD use "vendor" rules!

Vendor vs. SCMOS

- SCMOS wastes extra space for drawing the transistor.
- Area is money, so fabrication facilities resorted to creating rules based on distance and not grids!
  - Grids are still used to help keep connections together.
  - It also helps with contacting two transistors together.
- Vendor rules use absolute distances to draw transistors.
  - Very hard to do in a reasonable amount of time without some practice.
  - Almost all current technologies use vendor or manufacturer rules.
- SCMOS rules use grids based on lambda
  - This is our preferred method for this class.
  - Both methods are based on same principle of using design rules.

Overview of Physical Verification

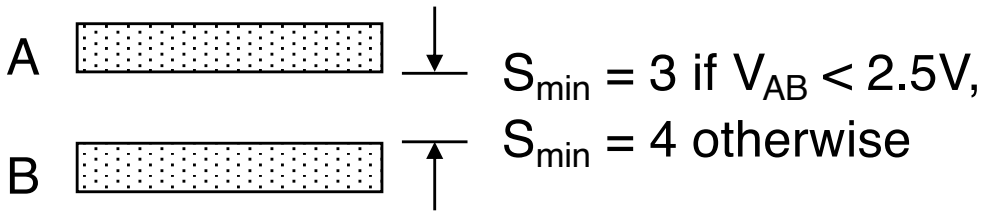
- **What is Physical Verification (PV)?**
- **General PV topics**
  - Design Rule Check (DRC)
  - Logical Versus Schematic (LVS) !!!
  - Verification Algorithms
    - Flat and Hierarchical (different file structures)
- **Approaches**
  - DRC
    - Place and Route, Flat and Hierarchical
  - LVS (some places can fire you if you forget!)
    - Place and Route, Flat and Hierarchical

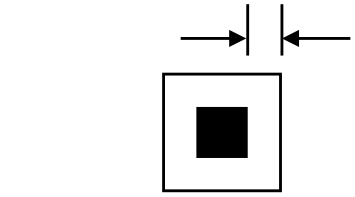


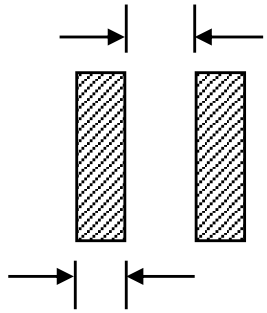
Can DRC Checkers use “the cloud”?

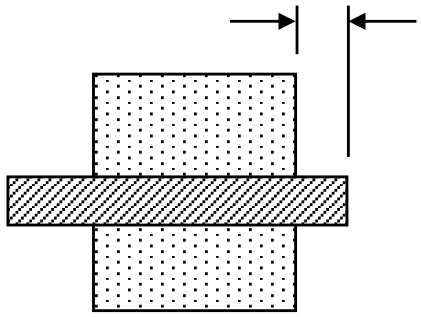
What is Design Rule Checking?

- Verification that layout geometry is legal
  - obeys set of *design rules*
  - minimum widths and spacings
  - extensions, overlaps
  - circuit-dependent rules
- Goal
  - verify that all rules are met
  - highlight places that rules fail and why
  - use minimum CPU time, memory
  - integrated DRC + layout editor
    - use existing data structures
    - check incrementally









Design Rule Checks (DRCs)

Goals:

- Manufacturability
- Yield

Analysis Inputs:

- **Foundry**
  - Rules
- **Design data**
  - Mask data, Layer information

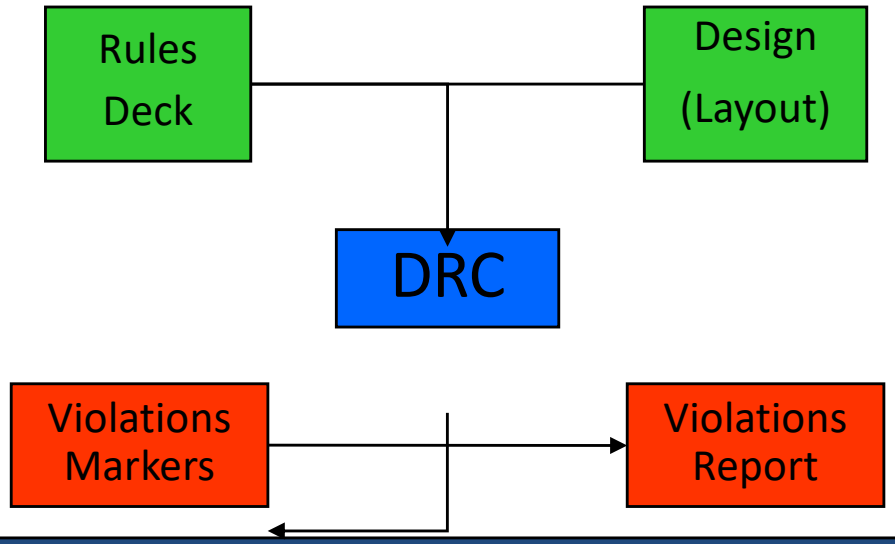
Typical checks performed:

For Manufacturing

- Width, Spacing, Minimum Area, Enclosed Area, Overhang, etc.

For Yield (repeating every **working** chip)

- Antenna, Electromigration, Latch-up, Electrostatic Discharge, Density



Design Rule Waivers

- Well tested special structures
  - Memory macros → almost always violates
- Special permissions with the cost of reduced yield
  - Antenna rules
  - Density rules
  - EM rules
- Many Intellectual Property (IP) vendors who supply silicon or instances (instantiations) of IP that they have worked out with the fabrication facility of "fab" with certain errors that have waivers.
  - That is, they are allowable, but they may or may not decrease yield.
  - Design Rules are for making sure yield is maintained and that the chip becomes what it is supposed to.

Whew!

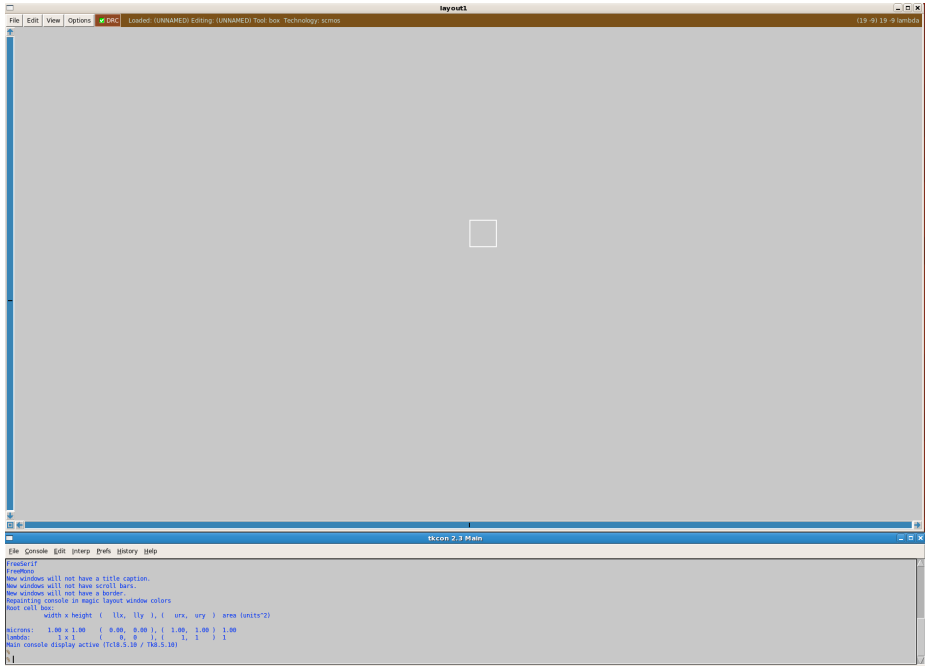
- Boy, that was tough!
- Seems like an enormous workload for any user!
  - Let's make it easier!!!!
  - Boy, this will be a common trait – if you can make it better and easier, do it!
- Design Rule Check or DRC
  - Let the computer check it for you!
  - The question is how to get your system to enter a layout!

Magic : the layout editor!

- Magic was a tool created by John Ousterhout who was previously known for inventing a language called Tcl/Tk (pronounced Tickle)
  - John now runs Sun Microsystems or has so for the last couple of years.
- This tool was invented for easy layout and is built on Tcl/Tk
  - Its further refined by a friend of mine, Tim Edwards
  - http://www.opencircuitdesign.com
- Magic is simple to learn:
  - Source startup script!
  - Type “magic”

Magic Environment

- Magic environment has two windows : the text and graphical window.
- I like to have my text and window windows separated and visible, because you only see graphics in the graphics window and you only see text in the text window – confusing?
  - Let me show you an example!



Mouse

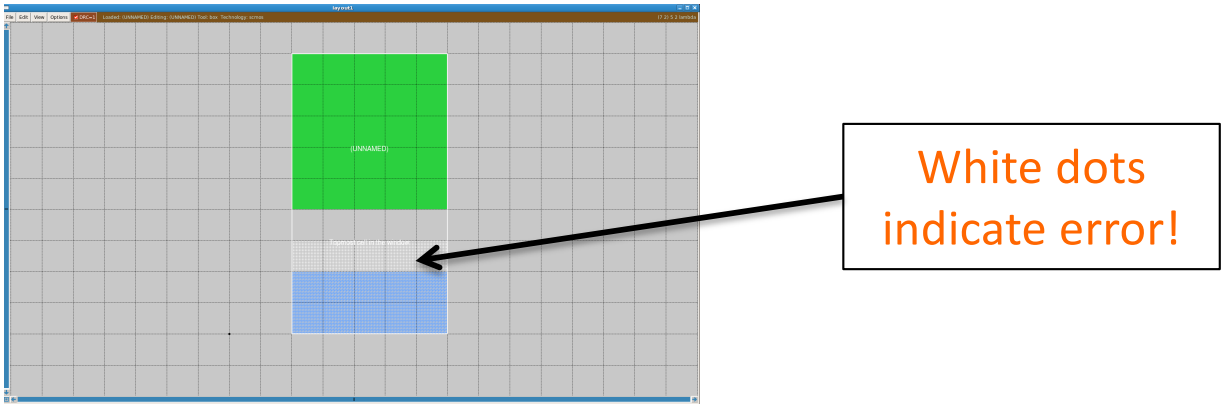
- The mouse draws a box!
- If you don't have a box, type "tool box", but it should run by default
- Buttons:
  - Left Button → lower left hand part of box
  - Right Button → upper right hand part of box

Menus

- All menus are invoked with the GUI or by typing the command
- Its best to always use the command
  - Magic uses the : to prefix the command.
- Example:
  - :paint ndiff
  - :paint poly
  - :paint metal1
  - :paint m1

DRC : Design Rules Check

- DRC or design rules check is used to tell if there are errors with the SCMOS rules.
  - In our case SUBM_SCMOS
- There are two types:
  - Interactive : tells error based on drawing
  - Batch-based : you have to run a command to show the errors (by far, the most common)
- Luckily Magic uses interactive DRC which is supppperrr cool!



Grids

- Since each grid is a lambda, its important to keep the grids at 1 when starting at.
- They should default to 1, but you can change by typing: “grid 1”

Saving

- Saving is a troublemaker
- In the past, the old magic tool used to kill everything, and it did this often
- However, I believe the current version does have some backup features.
- But get in a good habit of keeping backups
  - Since files tend to copy old files, its important to keep the filename different
- Example
  - :save james
  - :save backup1gopokes
  - :load james

Hierarchy

- As stated earlier, everything is a hierarchy
- However, its good to get in the habit of saving files into the hierarchy.
  - For example, I will typically work on an inverter in one directory.
  - Then, make another directory for the hierarchy above it and work there.
  - Then, copy files from the lower-level hierarchy into the other.
- Saving files is highly recommended!!!!!!!
  - Some students in the past would run scripts to save files periodically
- To use a hierarchy, type ":getcell inv" where inv is the lower-level cell.
  - You will see a box to represent the hierarchy
  - To view it, type "x"
  - To un-view it, type "Shift x"

Moving

- One of the coolest features of Magic is the ability to move blocks.
- This can be done with the select key and the movement keys.
- To select layout, highlight a box and type “a”
- To move a block, use QWER
  - Q = left
  - W = down
  - E = up
  - R = right
- Use interactive DRC to help make sure things are DRC free ☺!

Editing

- Some magic commands are not cool!
- For example, editing and resizing blocks is hard.
- Its advisable to find what methodology works best for you.
- Here are some strategies:
  - Deleting cells and starting over
  - Saving files and editing Magic file manually (Magic is saved as a text file).
    - This is extremely useful for text placement of pins
  - Opening up last saved file and starting again

Concluding Remarks

- Please see http://stineje.github.io
  - Click Magic and try tutorial!
  - More importantly, try logging onto our ECE server so you get a head start on future work!
- Make sure you know how to set up your VNC session!
- If you have questions, ask on piazza, class, or in my office!