Spiral 2-8

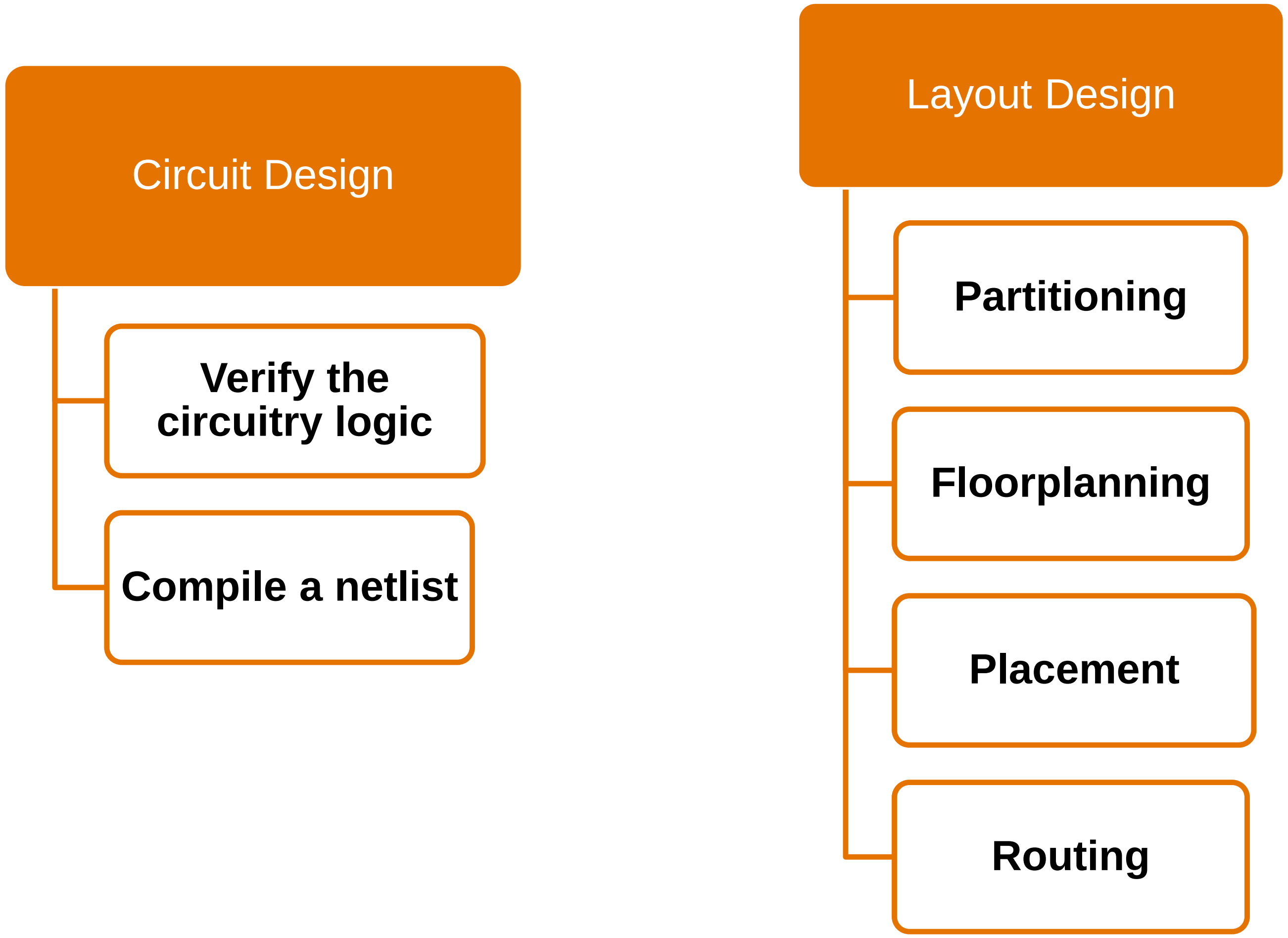
Cell Layout

Learning Outcomes

- I understand how a digital circuit is composed of layers of materials forming transistors and wires
- I understand how each layer is expressed as geometric patterns called masks
- I understand the difference between custom and standard cell ASIC flow
- I understand how FPGAs implement arbitrary logic designs by producing a bit stream which is the contents of Look-Up Tables and mux selects
- I understand the pros and cons of ASIC vs. FPGA design targets

LAYOUT

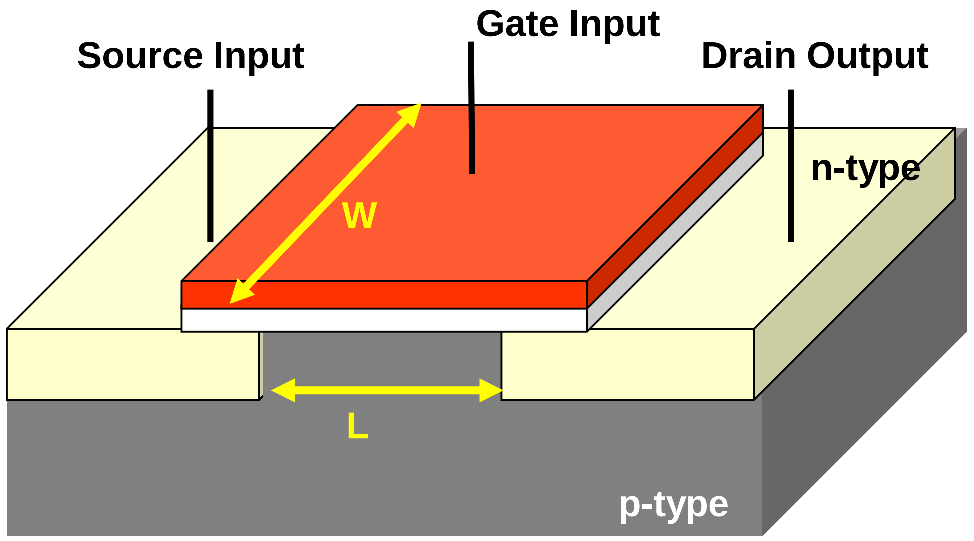
Digital Design Overview

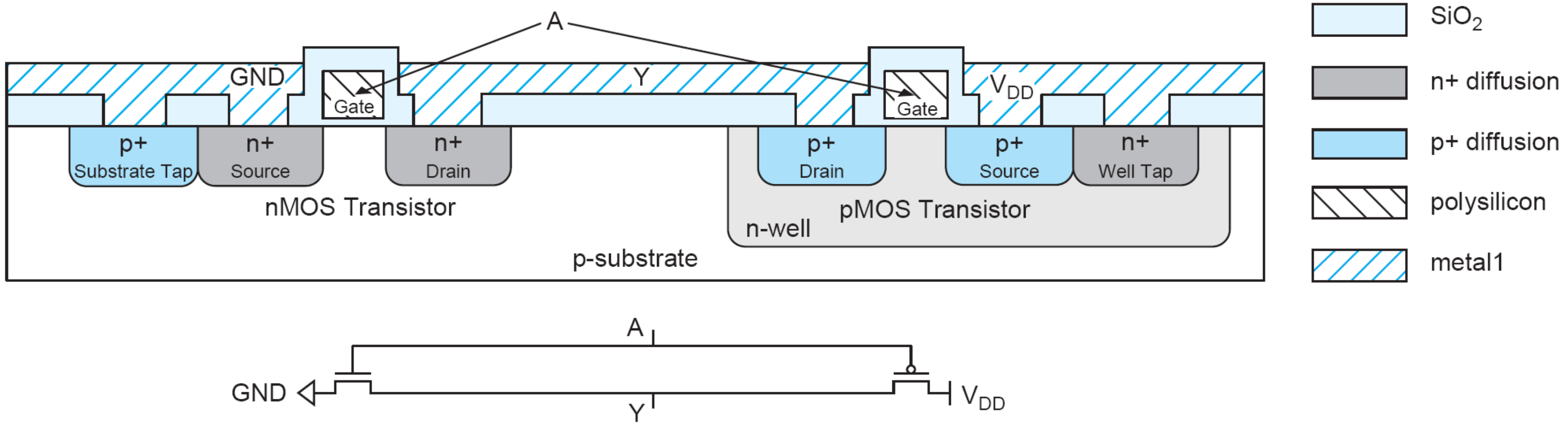


- A flowchart of digital
  - From concept through testing and finally to a layout

Layout

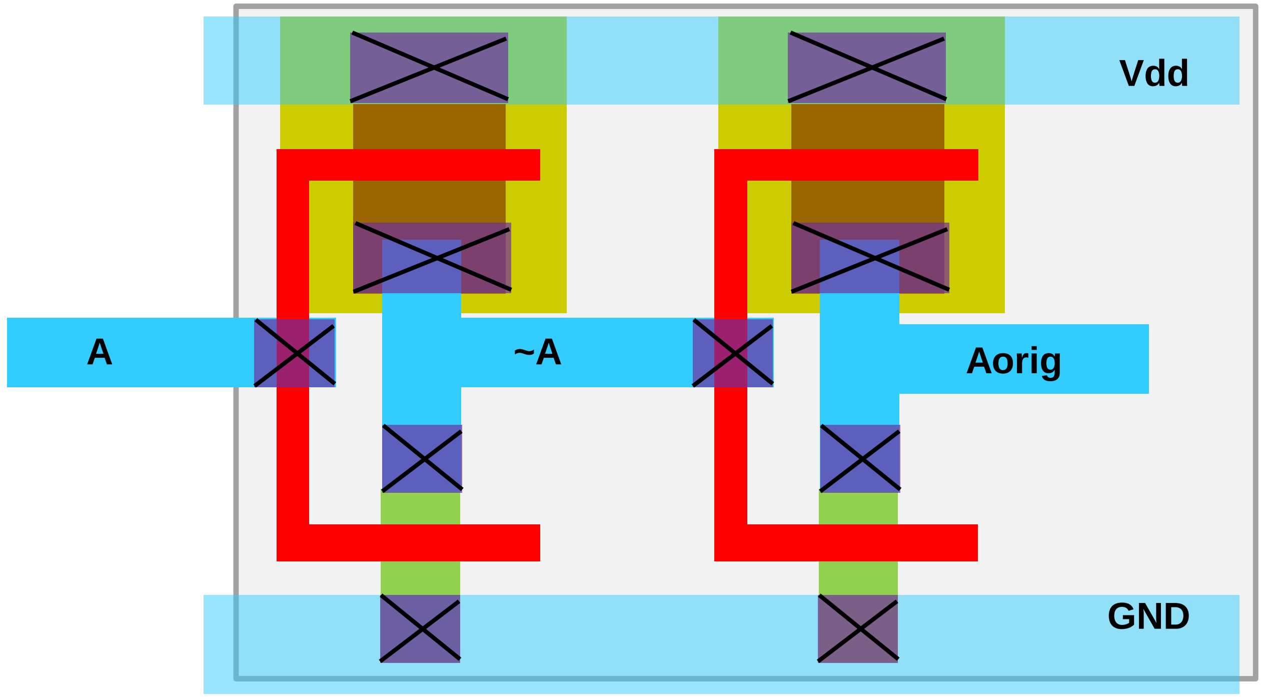
- We need to describe the patterns of material to deposit and build on the silicon surface
- We will draw it from a top-down perspective

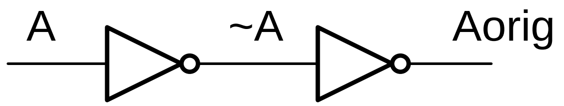


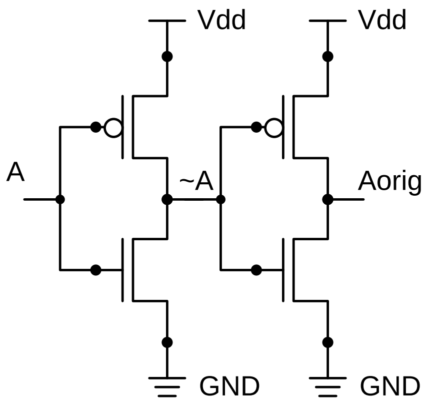


Layout

- Below is a notional layout of two inverters back to back
- Let's go through the steps for how we would lay this out

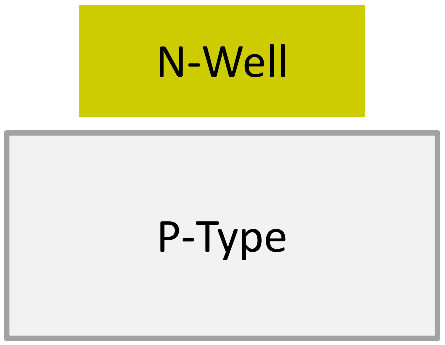


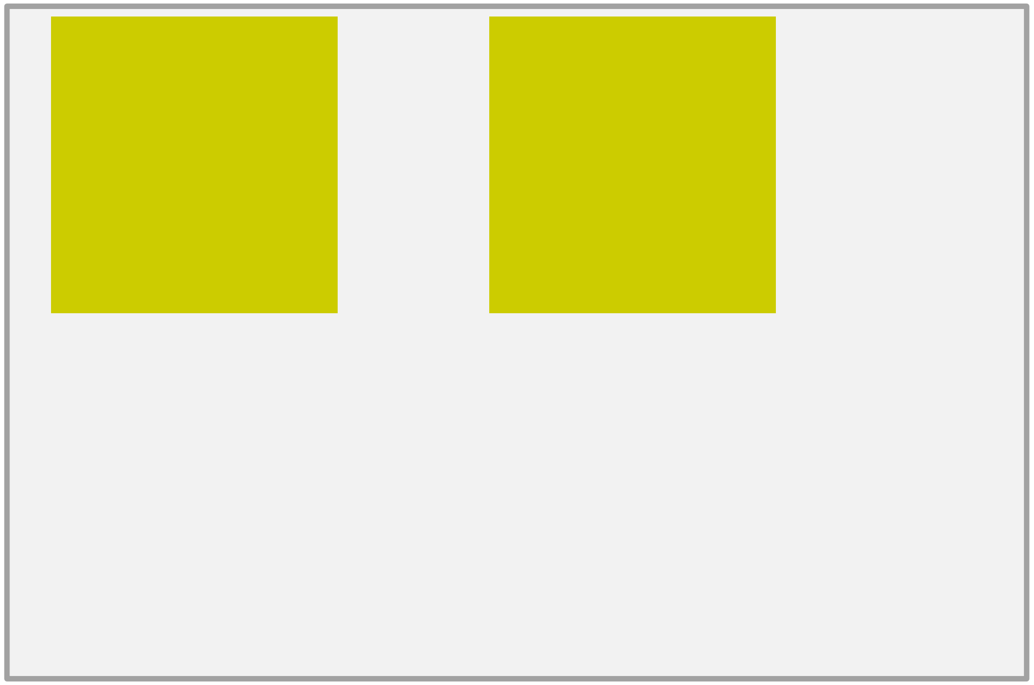


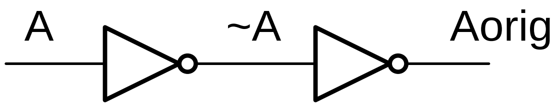


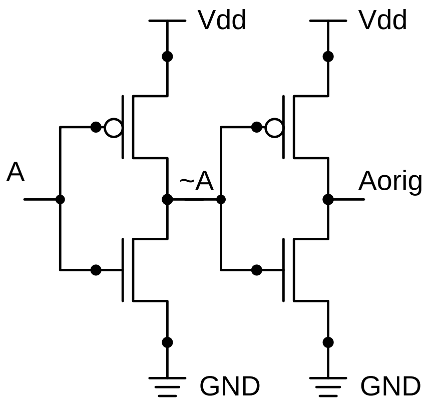
Layout

- Start with p-type substrate which is what we need for NMOS
- Add n-well area for the PMOS transistors







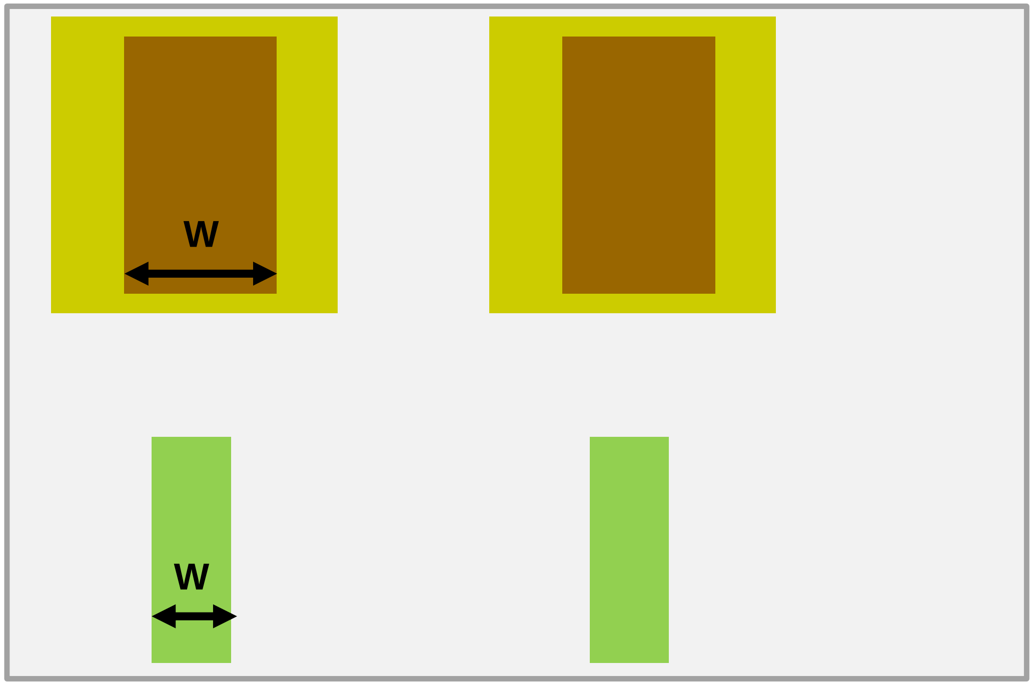


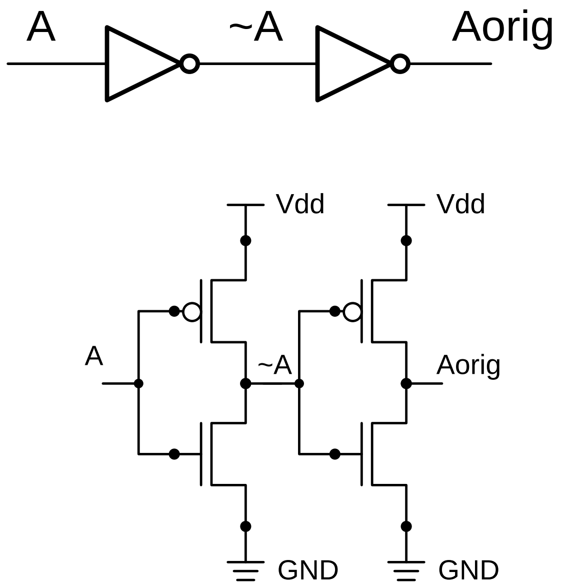
Layout

- Now we lay down the n- and p- diffusion areas for the source and drain of both NMOS and PMOS
- Notice the W/L relationship
  - Right now the length seems longer than it really will be

p+ diff

n+ diff.

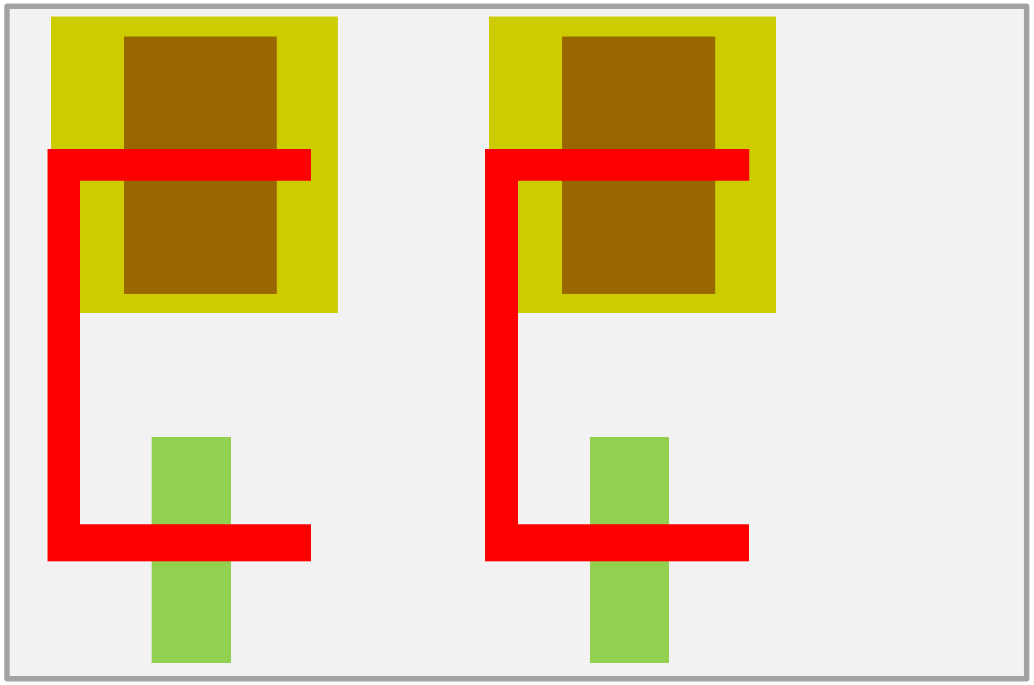


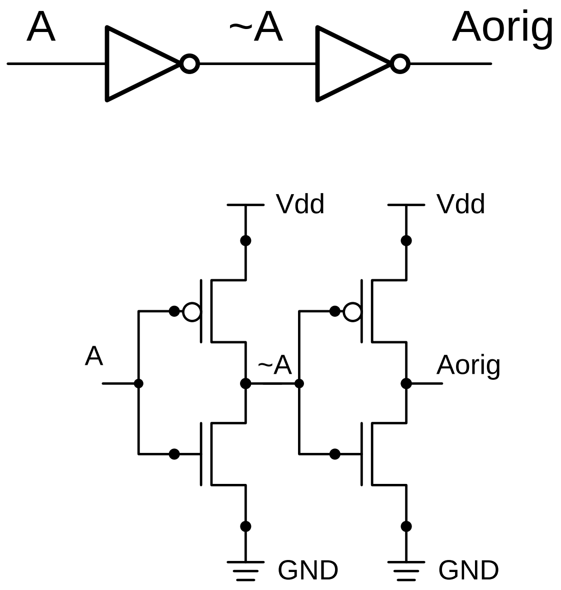


Layout

- Now we draw the **poly**silicon gates

Poly 🟥



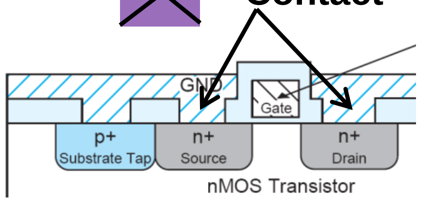


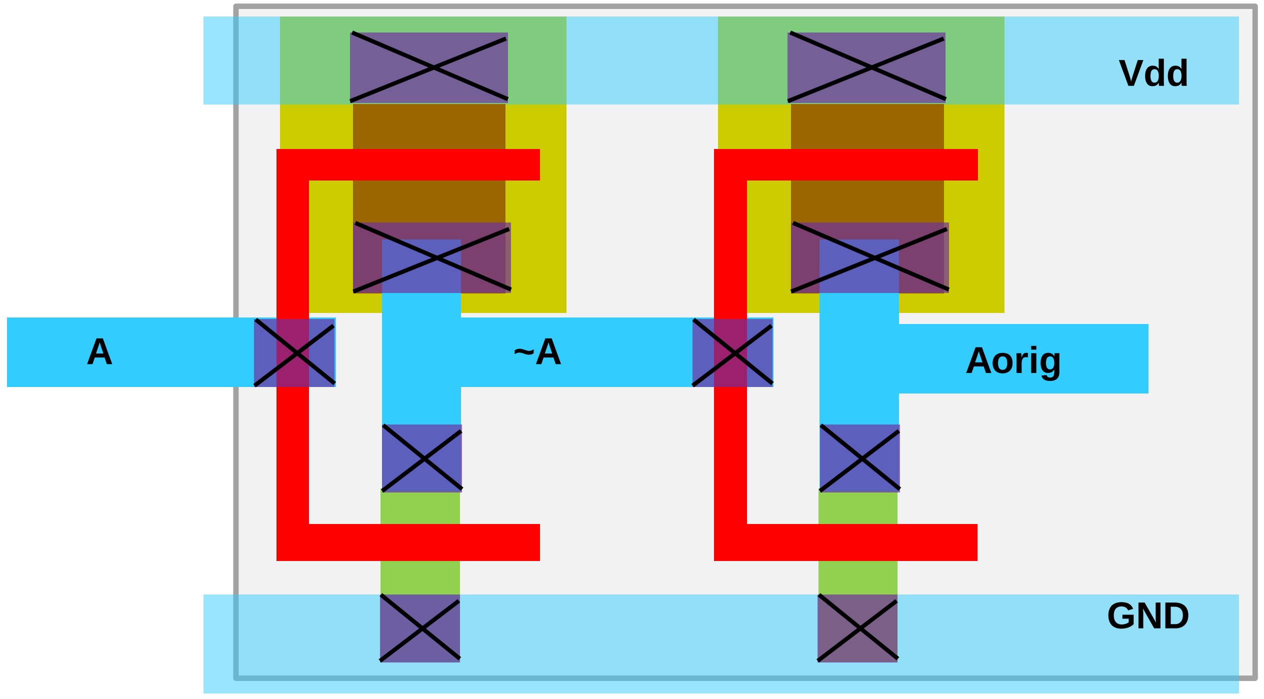
Layout

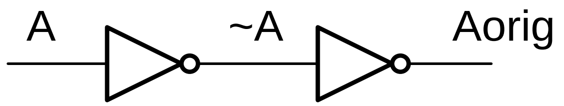
- Now we add the L1 metal (**M1**) wires which include the Vdd and GND rail as well as the input and output signals
- To make a connection from the M1 level to the surface of the chip we add contacts

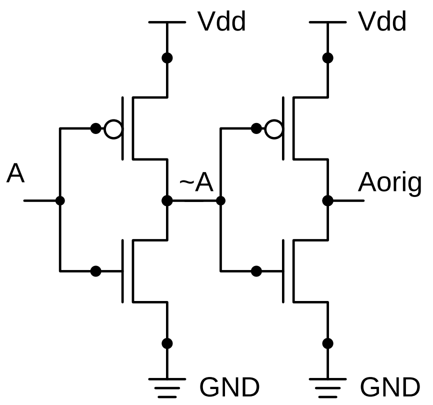
M1

Contact



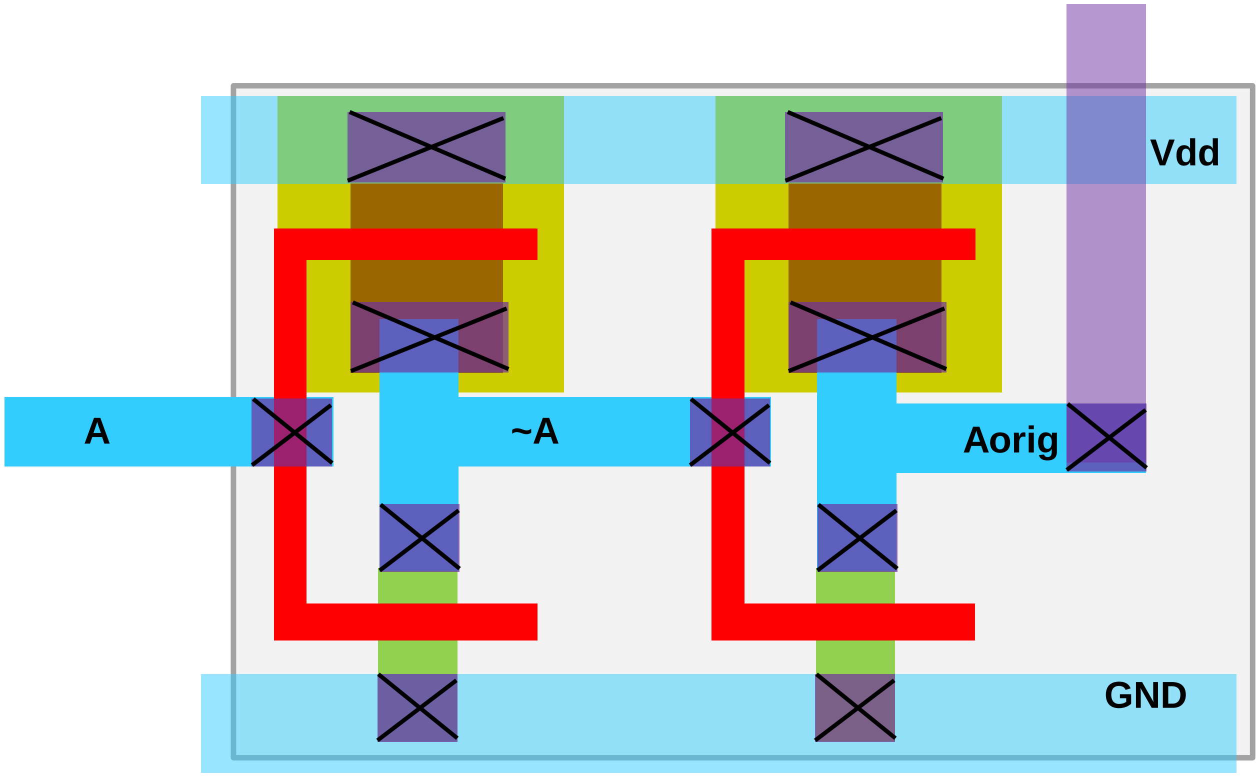


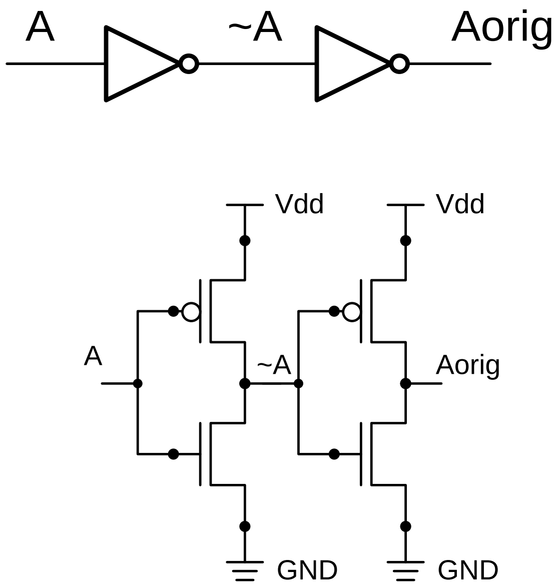




M2 – Mn

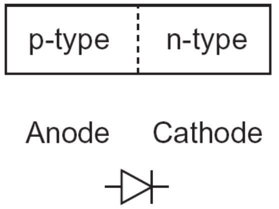
- How do we get Aorig out from between Vdd and Gnd?
- We need another level of metal, M2
  - Just like a freeway interchange

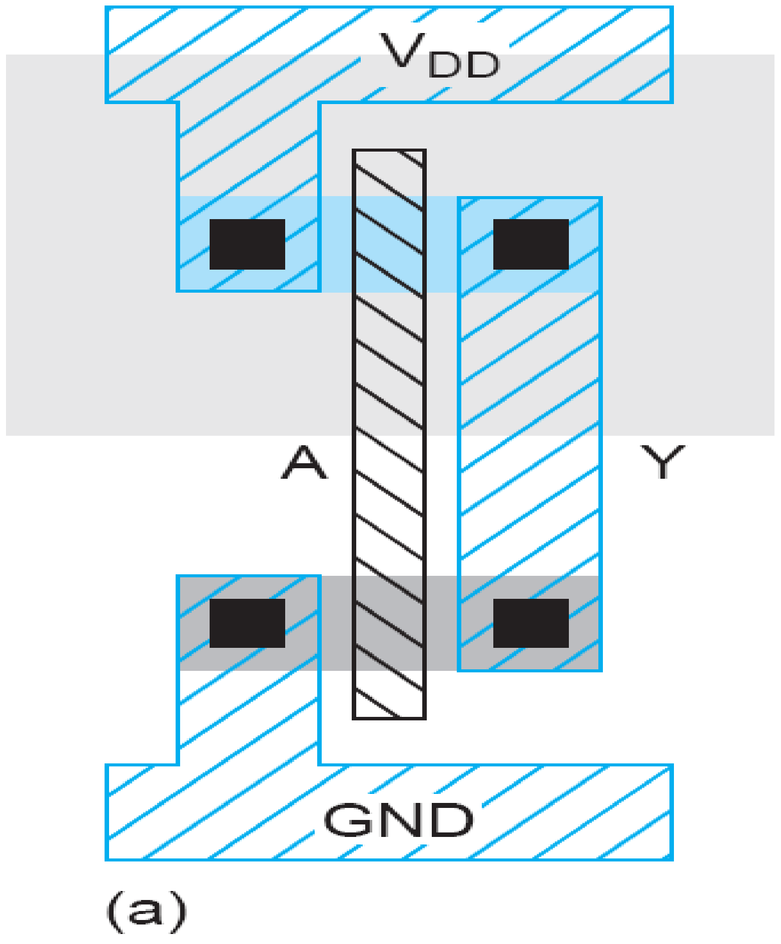


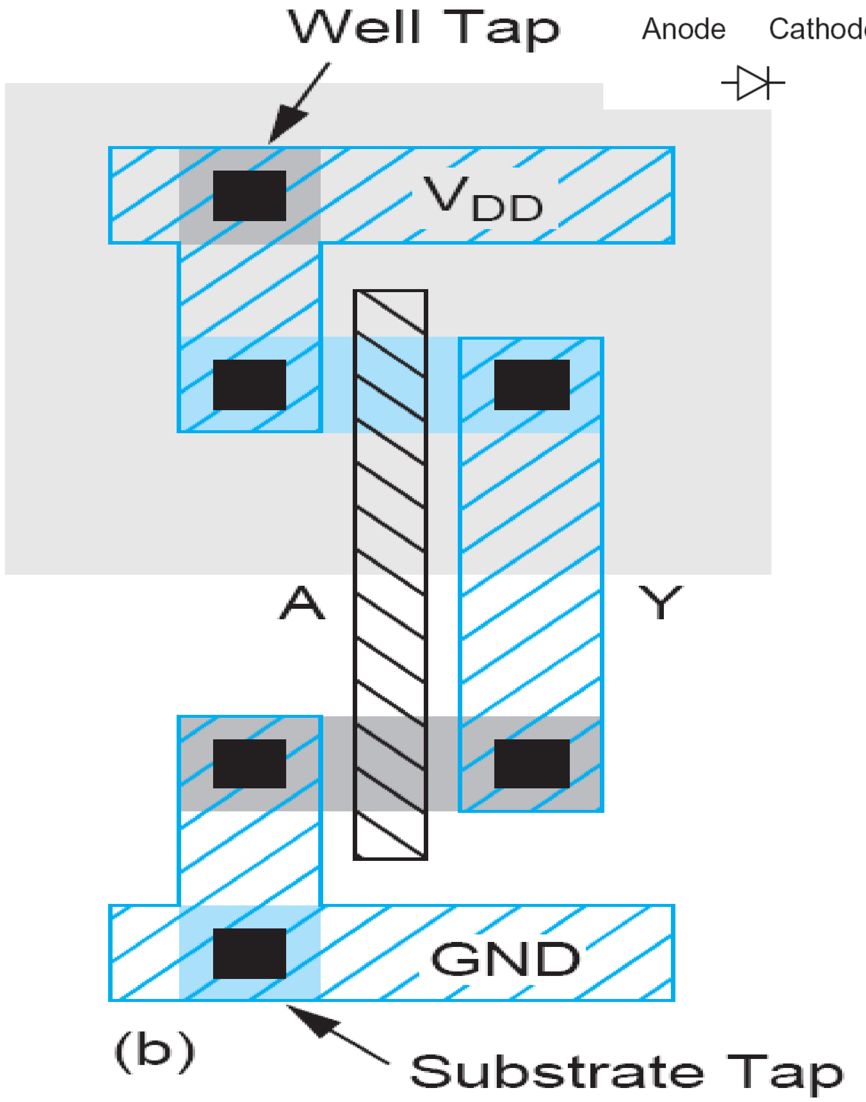


CMOS Inverter

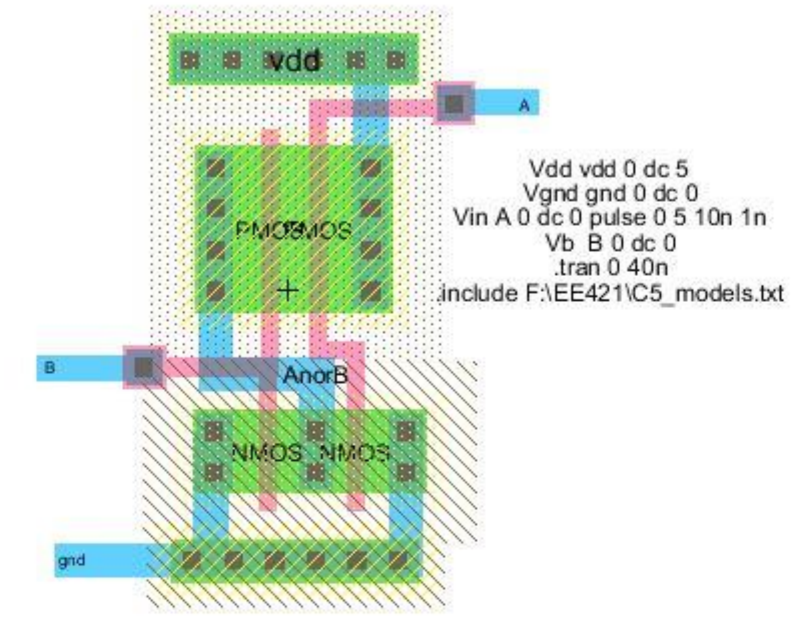
- Taps in (b) to connect n-well and p-substrate to V_{DD} and ground respectively



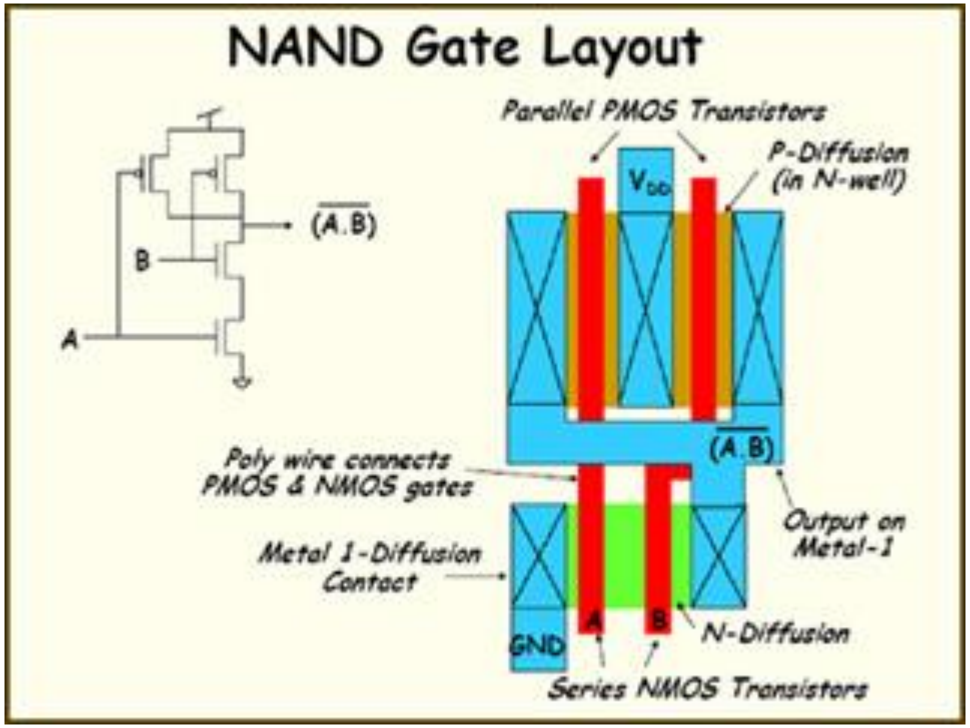




NAND and NOR Gate Layout



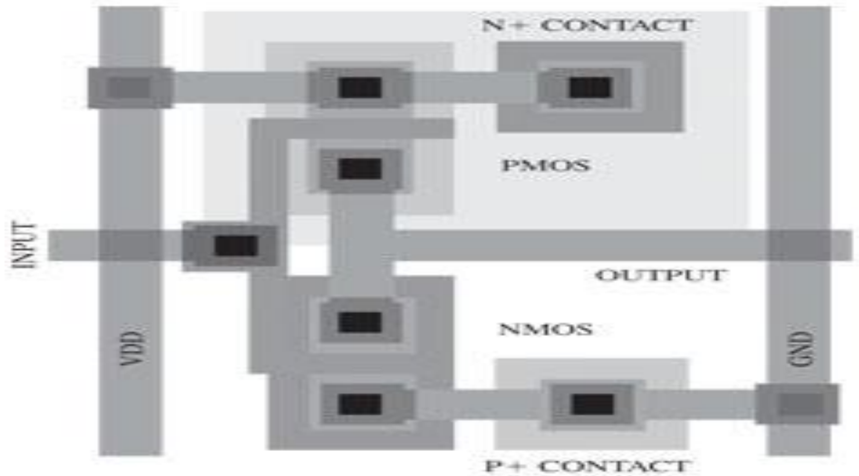
```
Vdd vdd 0 dc 5
Vgnd gnd 0 dc 0
Vin A 0 dc 0 pulse 0 5 10n 1n
Vb  B 0 dc 0
.tran 0 40n
.include F:\EE421\C5_models.txt
```



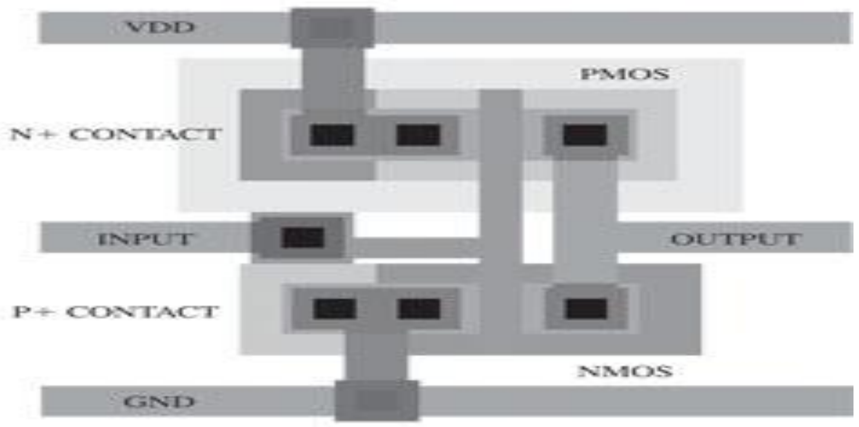
http://cmosedu.com/jbaker/courses/ee421L/f13/students/adamsk5/lab6/nor_lay_sim.JPG

http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-884-complex-digital-systems-spring-2005/6-884S05.jpg

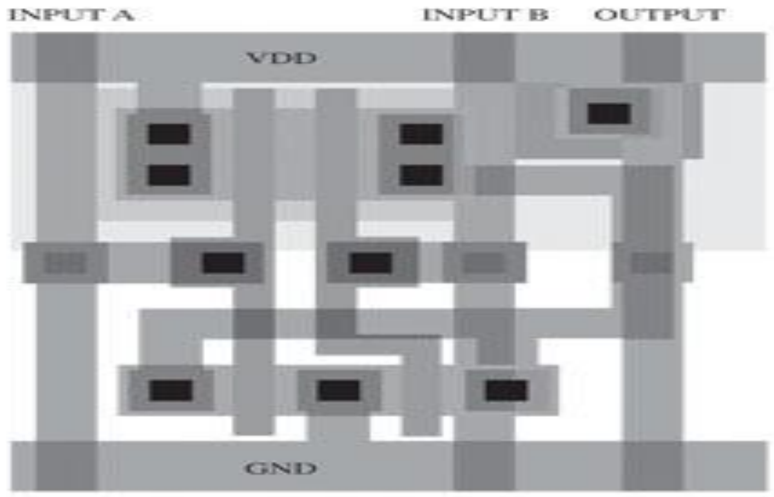
CMOS Gates – Inverter, NAND2, NOR2



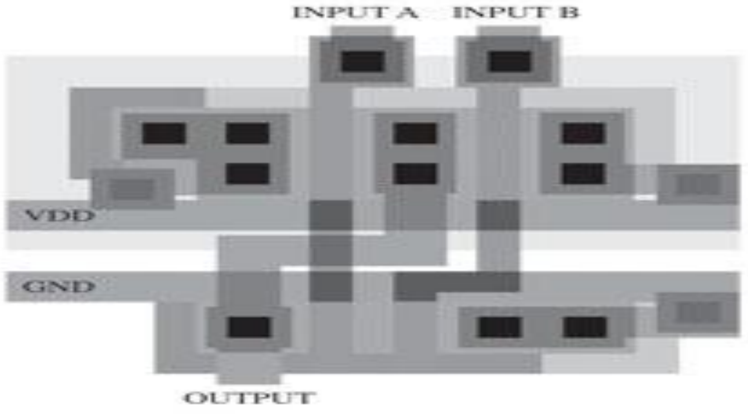
CMOS INVERTER (version 1)



CMOS INVERTER (version 2)

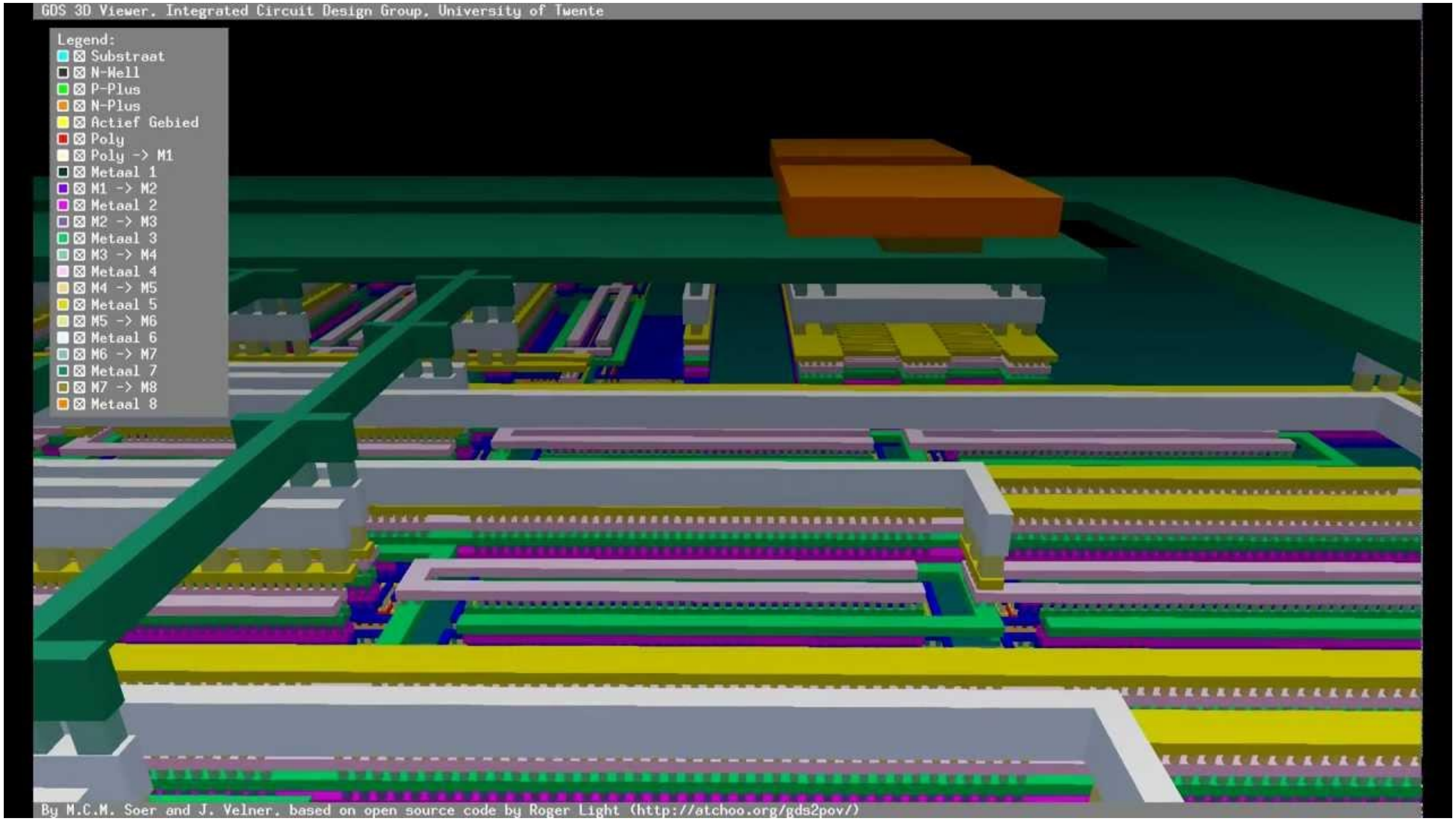


CMOS NOR2 GATE



CMOS NAND2 GATE

Chips Are 3D Sandwiches



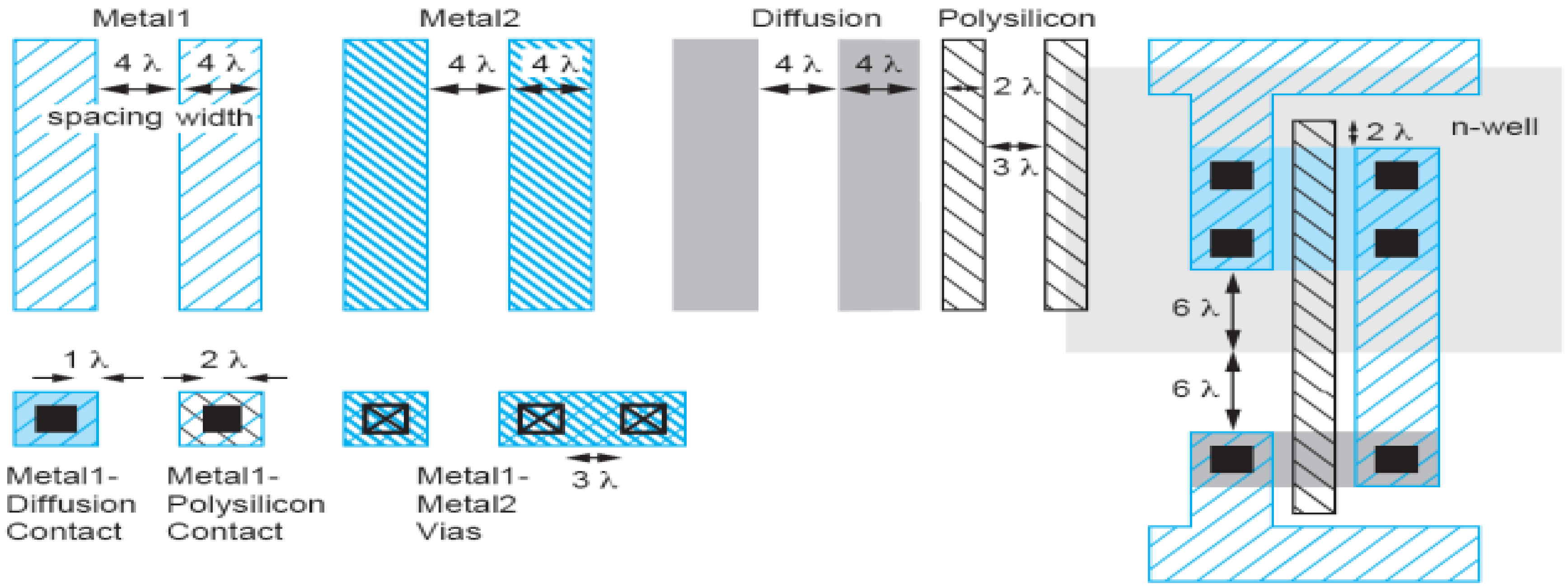
http://i.ytimg.com/vi/F4EArOqNNSU/maxresdefault.jpg

Ensuring we can fabricate a working chip

DESIGN RULES

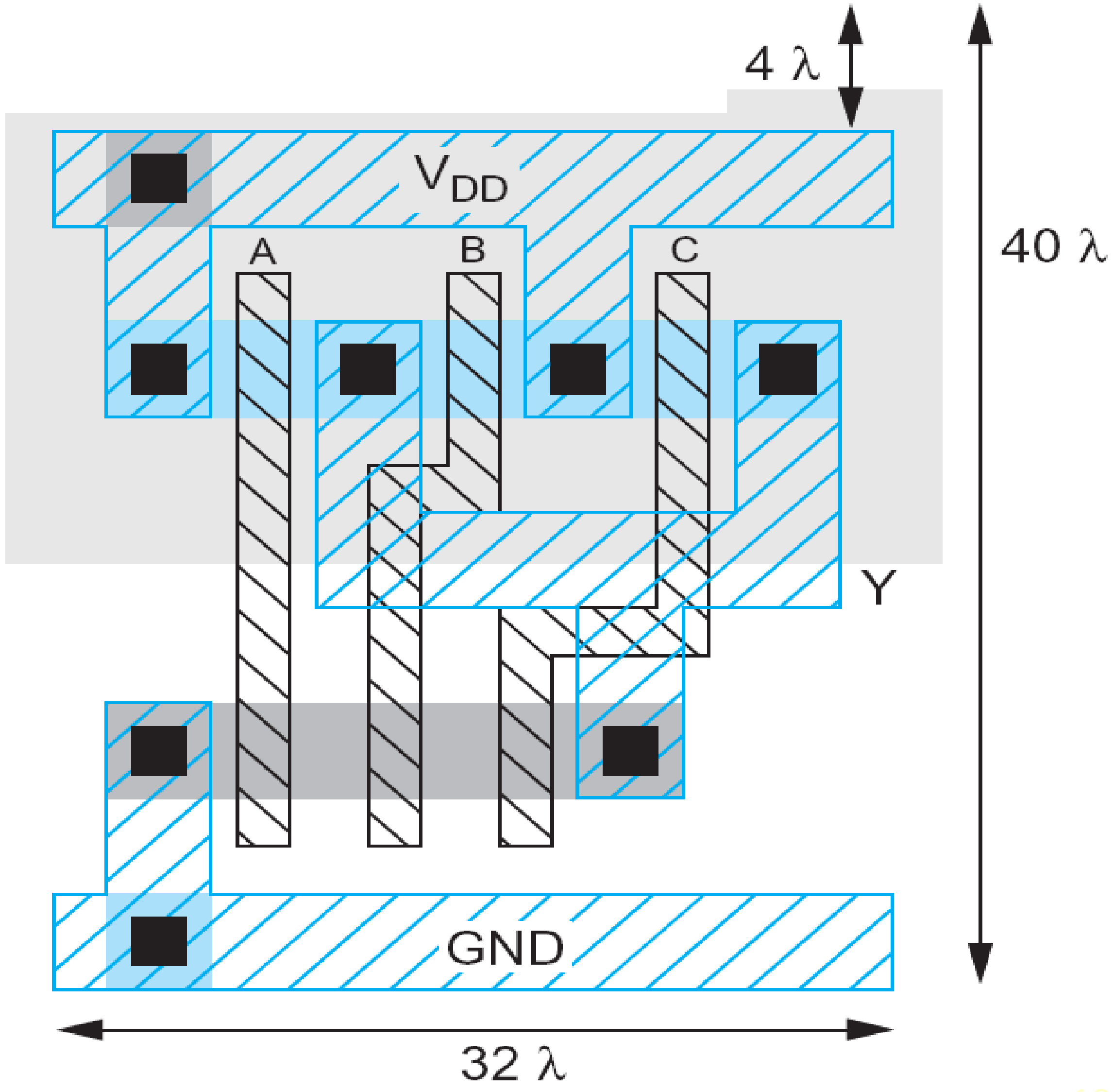
Reminder: Layout Design Rules

- Lambda Rules: One lambda = one half of the "minimum" mask dimension, typically the length of a transistor channel
- Lambda Rules are based on the assumption that one can scale a design to the appropriate size before manufacturing
  - Every spacing and sizing value is presented based on multiples of lambda
  - The layout tool has a design rule checker (DRC) to verify those. In case of any violations the tool will point that out



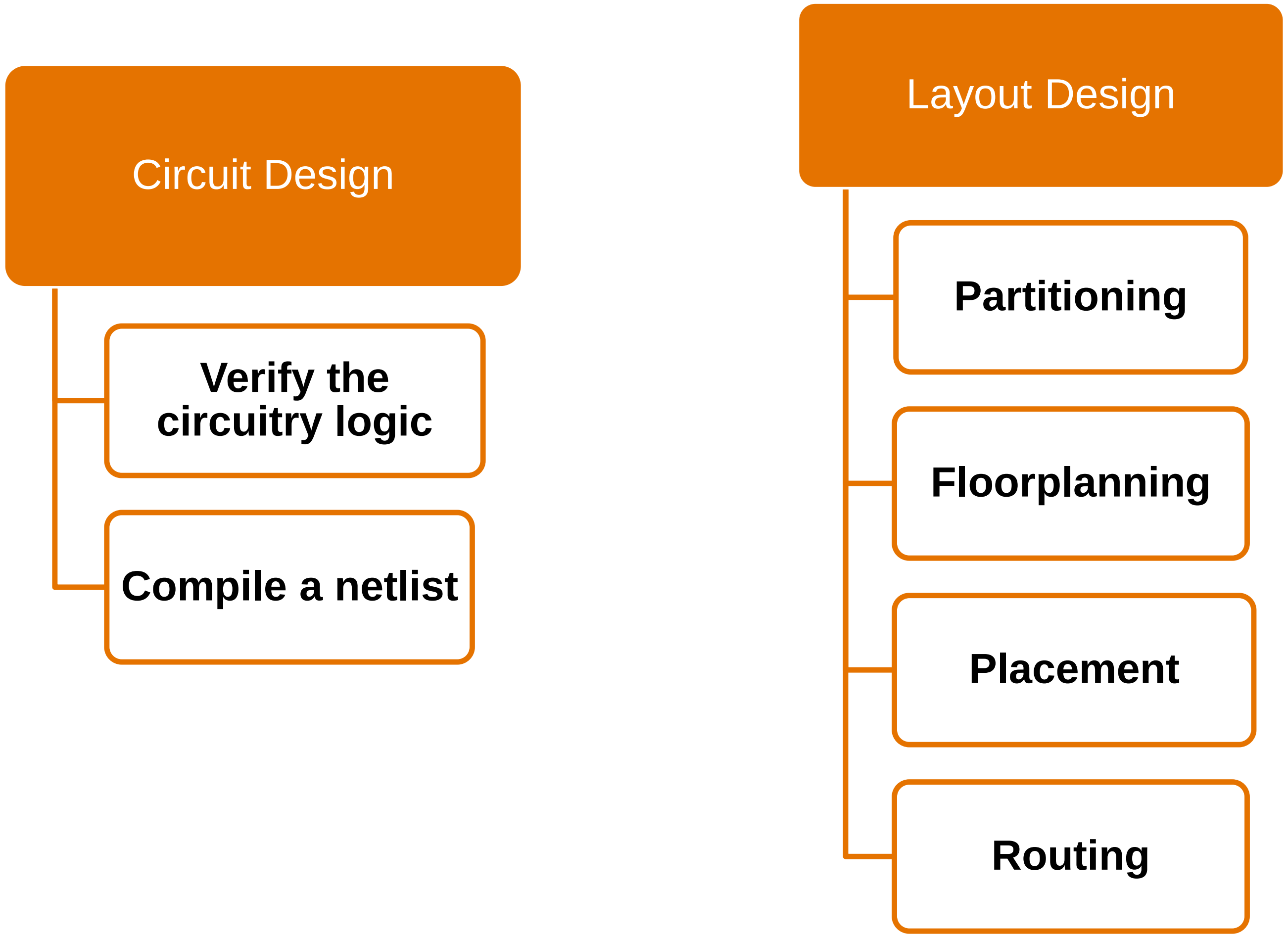
CMOS NAND3

- nMOS and pMOS transistors in series, and in parallel, respectively



CUSTOM OR STANDARD CELL ASIC (NOT FPGA) DESIGN FLOW

Digital Design Overview



- A flowchart of digital
  - From concept through testing and finally to a layout

Layout Design

- After circuit design, we obtain a netlist which could be easily translated into a schematic. Now, let’s start the layout design (a.k.a. physical design)
- **Partitioning**
  - Divide the chip into smaller blocks
  - This is done based on some goals and constraints, e.g., to minimize the number of connections between the blocks, but in general it is done to separate different functional blocks and simplify their physical design process and also to make placement and routing easier
    - For example, during partitioning you may decide to divide your design into two blocks, such that the total number of connections between the gates in different blocks is minimized

Layout Design (cont.)

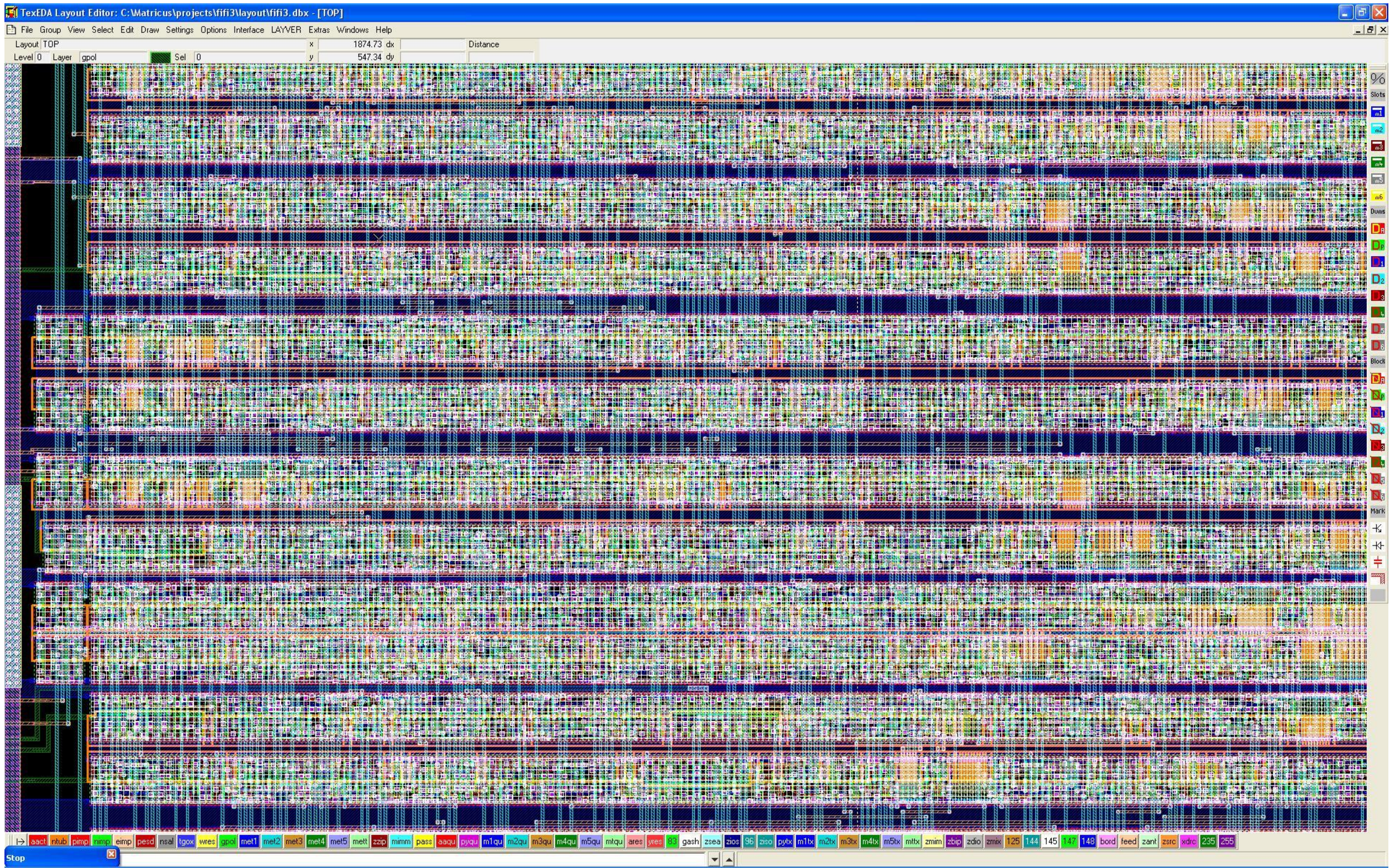
- Floorplanning
  - Create functional areas for your chip. For example, decide where to place FPU (Floating Point Unit), RAM, MPU (Microprocessor), ROM on the chip
  - Place the input and output (I/O) cells of your chip
  - Connect functional blocks with I/O pads or with each other
  - Check whether long wires would slow your design
- Placement
  - Nail down the exact positions of all logic gates within each block
  - Place I/O drivers
  - Similarly to other steps, placement is done based on goals and constraints, e.g., such that the total approximate wire length is minimized

Layout Design (cont.)

- Routing
  - Route power nets and clock nets first. They are critical nets
  - Route rest of the nets
  - Routing is typically done in two steps of global routing and detailed routing. In global routing the resource (channels) for the wires are selected and in detailed routing, the wires are assigned to a specific routing track (metal layer) in the selected channels

Macro View of a Chip

- Place and route are very important aspects of design


TexEDA Layout Editor: C:\Matricus\projects\fifi3\layout\fifi3.dbx - [TOP]

Layout TOP
Level 0 Layer gpol Sel 0
x 1874.73 dx
y 547.34 dy
Distance

File Group View Select Edit Draw Settings Options Interface LAYVER Extras Windows Help
Slots
Dums
Block
Mark
aact ntub pimp nimp eimp pesd nsal tgox wres gpol met1 met2 met3 met4 met5 mett zzip mimm pass aaqu pyqu m1qu m2qu m3qu m4qu m5qu mtqu ares yres 83 gash zsea zios 96 ziso pytx m1tx m2tx m3tx m4tx m5tx mttx zmim zbip zdio zmix 125 144 145 147 148 bord feed zant zsrc xdrc 235 255
Stop

Standard Cell Library

- Standard Cell Library
  - Intellectual property provided by a vendor
  - Has many pre-defined gates (cells) that are already laid out at various sizing levels for drive strength (to achieve desired delay)
  - Design Flow
    - You develop a design
    - A synthesis software tool determines what cells are needed
    - A place and route tool determines how to place them and route the input/output of each cell appropriately (i.e. using various metal layers)
  - Example
    - http://web.engr.oregonstate.edu/~traylor/ece474/reading/SAED_Cell_Lib_Rev1_4_20_1.pdf

FPGAS

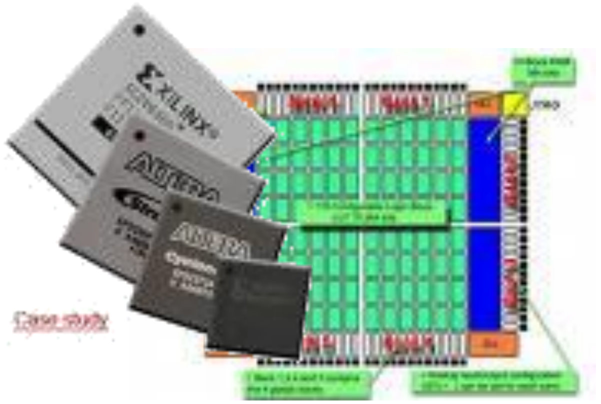
Progression of Logic Density

- Small Scale Integrated (SSI) Circuits
  - 1960's and 1970's
  - A few gates on a chip
- Medium Scale Integrated (MSI) Circuits
  - 1970's
  - Around a hundred gates per chip ('283s and '85s)
- Very Large Scale Integrated (VLSI) Circuits
  - 100's of millions of gates

Digital Design Targets

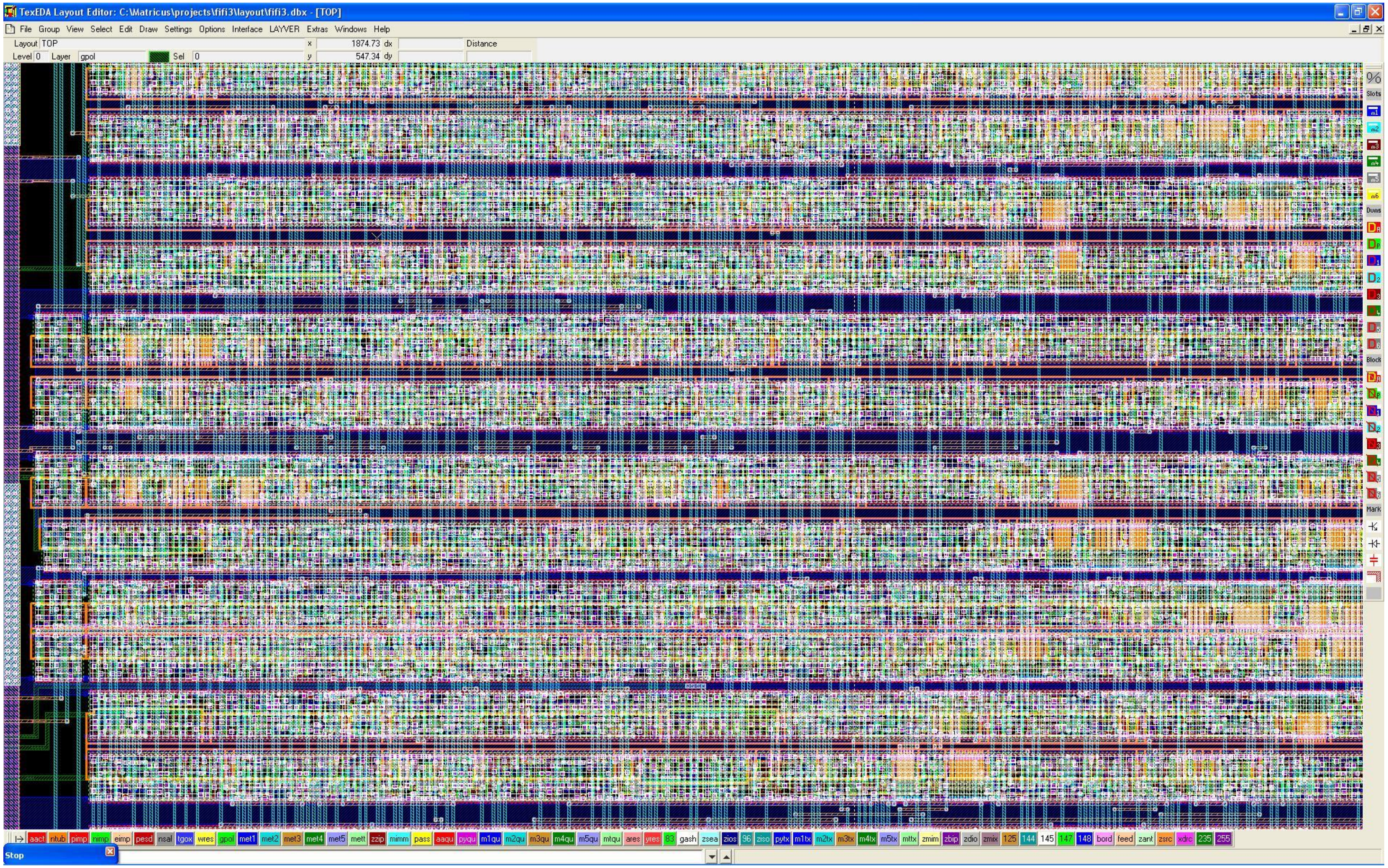
- Two possible implementation targets
  - Custom Chips (ASIC's = Application Specific Integrated Circuits): Physical gates are created on silicon to implement 1 particular design
  - FPGA (Field Programmable Gate Array's): "Programmable logic" using programmable memories to implement logic functions along with other logic resources tiled on the chip. Can implement any design and then be changed to implement a new one

In an ASIC design, a unique chip will be manufactured that implements our design at which point the HW design is fixed & cannot be changed (example: Pentium, etc.)



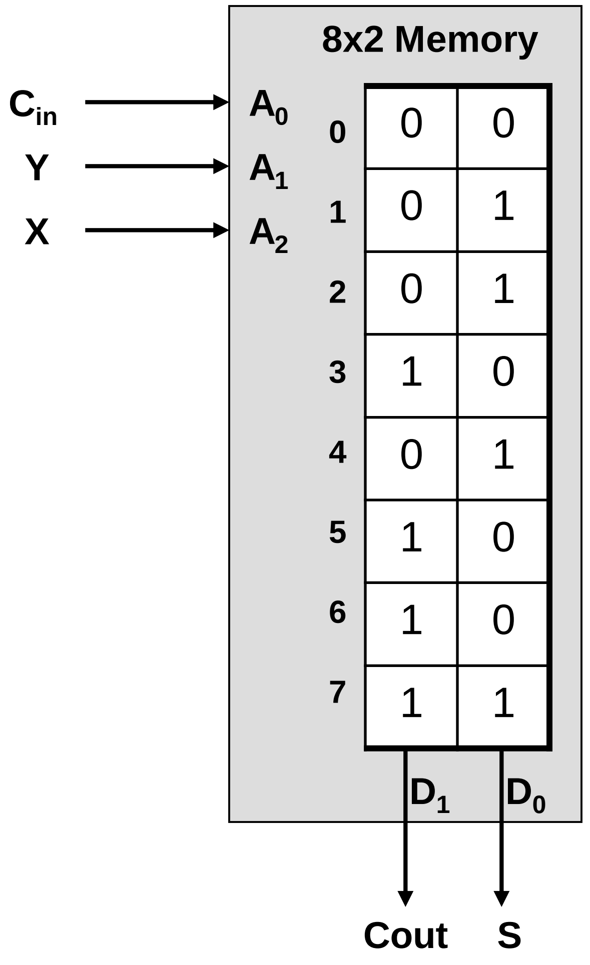
FPGA's have "logic resources" on them that we can configure to implement our specific design. We can then reconfigure it to implement another design

ASICs


TexEDA Layout Editor: C:\Matricus\projects\fifi3\layout\fifi3.dbx - [TOP]
File Group View Select Edit Draw Settings Options Interface LAYVER Extras Windows Help
Layout TOP
Level 0 Layer gpol Sel 0
x 1874.73 dx
y 547.34 dy
Distance
Slots
Dums
Block
Mark
aact ntub pimp nimp eimp pesd nsal tgox wres gpol met1 met2 met3 met4 met5 mett zzip mimm pass aaqu pyqu m1qu m2qu m3qu m4qu m5qu mtqu ares yres 83 gash zsea zios 96 ziso pytx m1tx m2tx m3tx m4tx m5tx mttx zmim zbip zdio zmix 125 144 145 147 148 bord feed zant zsrc xdrc 235 255
Stop

Basis of FPGA's

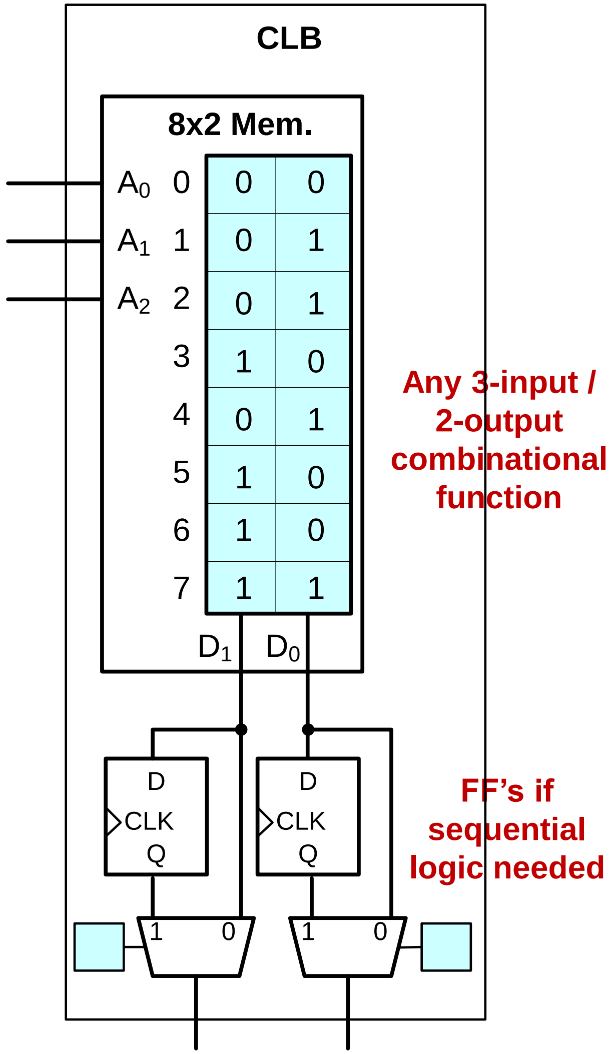
- Memories provide a universal way to implement a logic function
  - 2^n x m memory can implement a function of n-variables and m outputs
- If we use RWM (read/write memory) rather than ROM's we can change what function the memory implements
- Memories are referred to as Look-up Tables (LUT's)



Full Adder Implementation

Configurable Logic Blocks (CLB's)

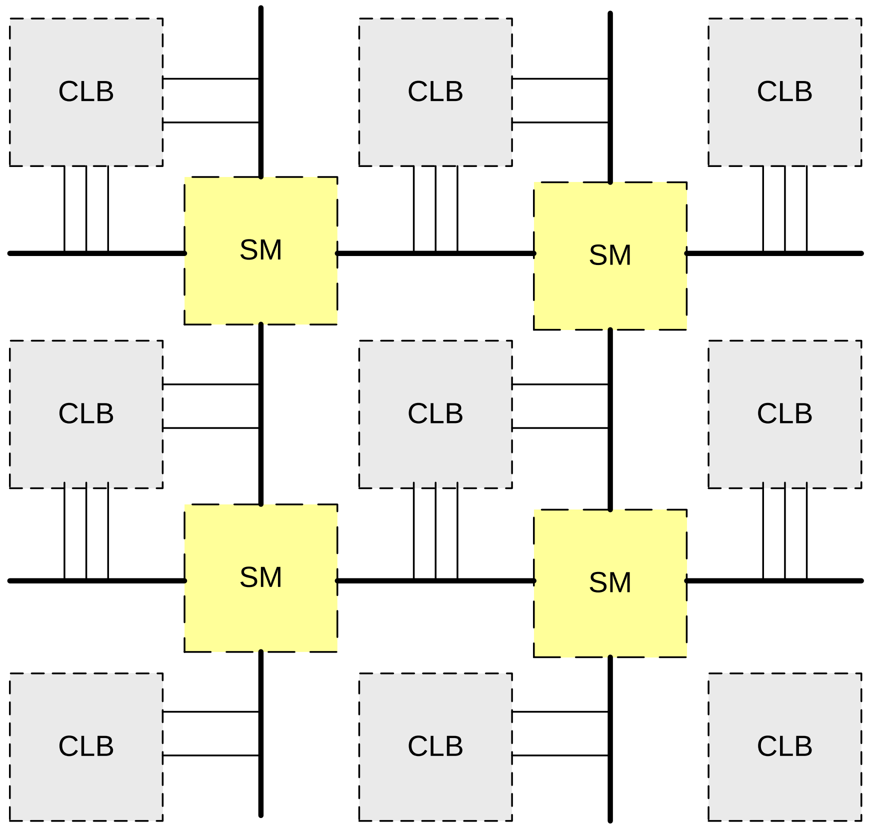
- Writable Look-Up Table
- D-FF's with bypass path
  - "Bypass" mux selects the pure combinational output of the LUT or the registered/D-FF output
- Blue boxes indicate programmable bits that control the operation and function of the logic


CLB
8x2 Mem.
A0
A1
A2
D1
D0
D
CLK
Q
D
CLK
Q
1
0
1
0
Any 3-input / 2-output combinational function
FF's if sequential logic needed

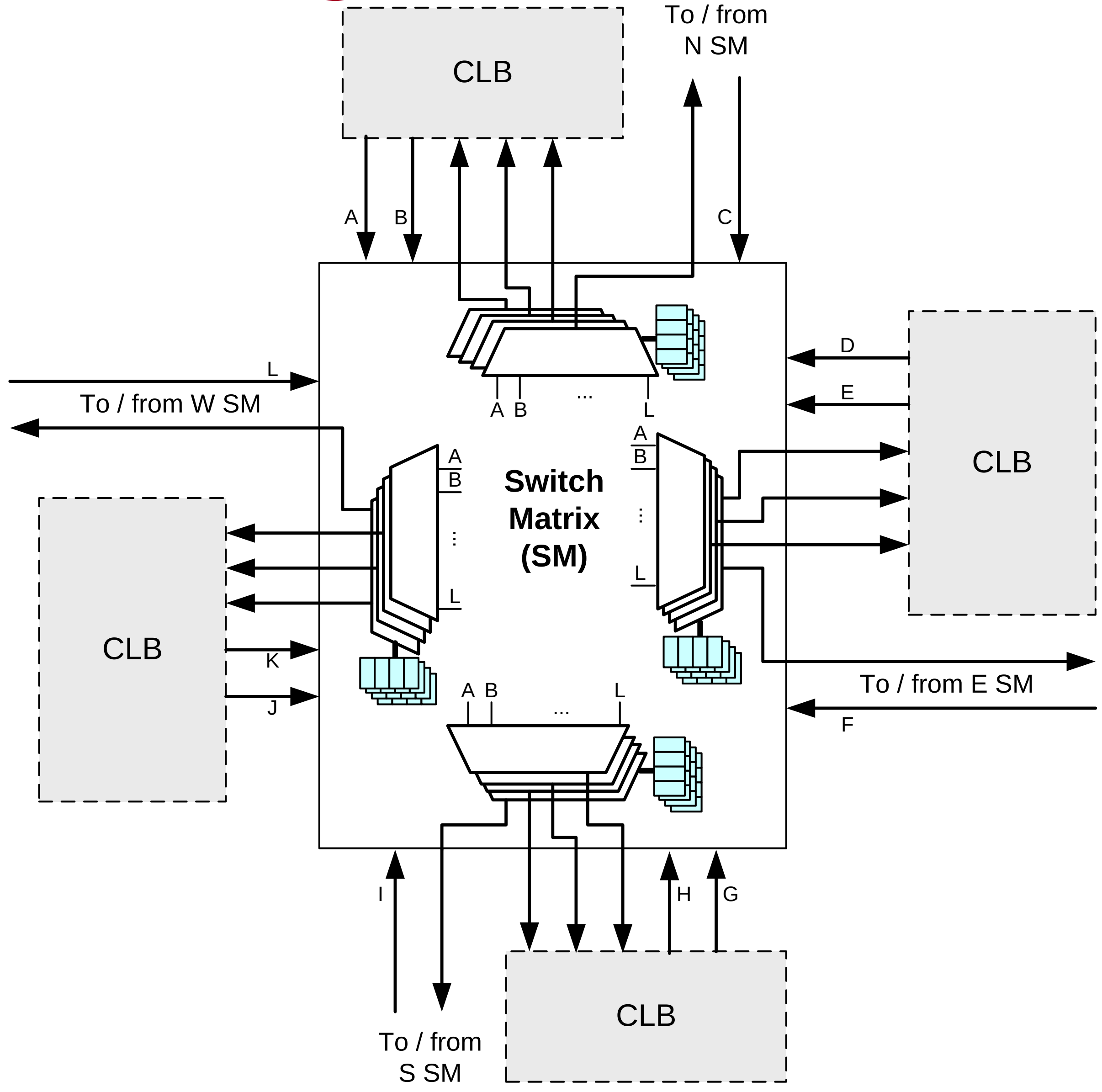
0	0	0
1	0	1
2	0	1
3	1	0
4	0	1
5	1	0
6	1	0
7	1	1

Routing & Switch Matrices

- Inputs and outputs of neighboring CLB's connect to a "switch matrix" (SM)
- Switch matrix is simply composed of muxes that allow us to "route" inputs and outputs to another CLB or further away

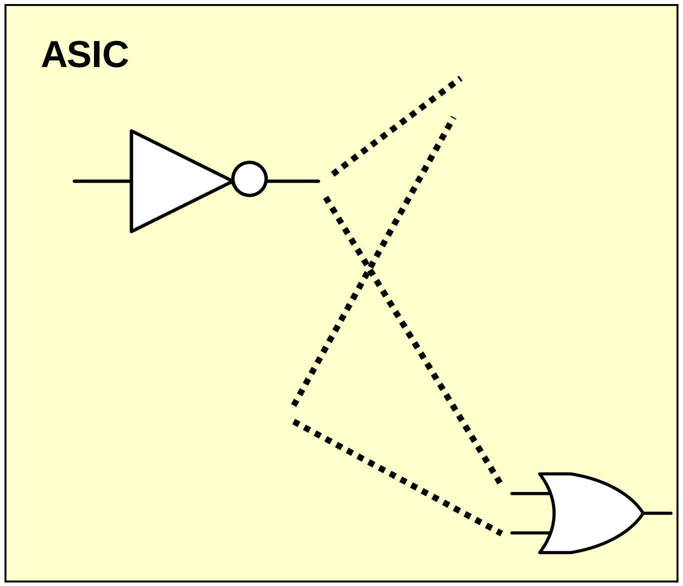


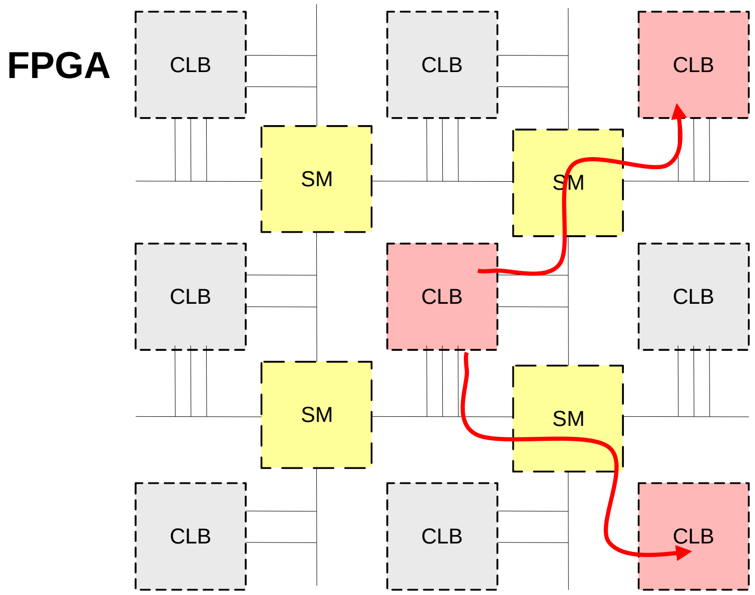
Routing & Switch Matrices



Place and Route

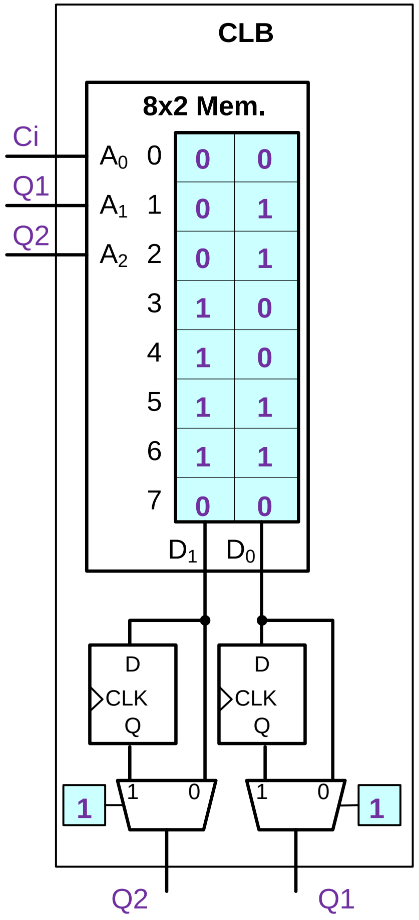
- **ASIC**: Find where each gate should be placed on the chip and how to route the wires that connect to it
- **FPGA**: Determine which LUT's should be used and how to route through switch matrices
- Affects timing and area
  - wiring takes up space and longer wires leads to longer delays

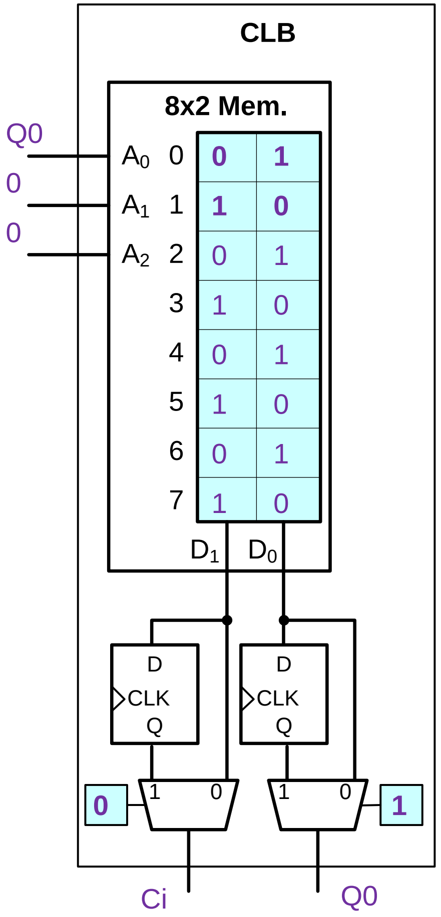


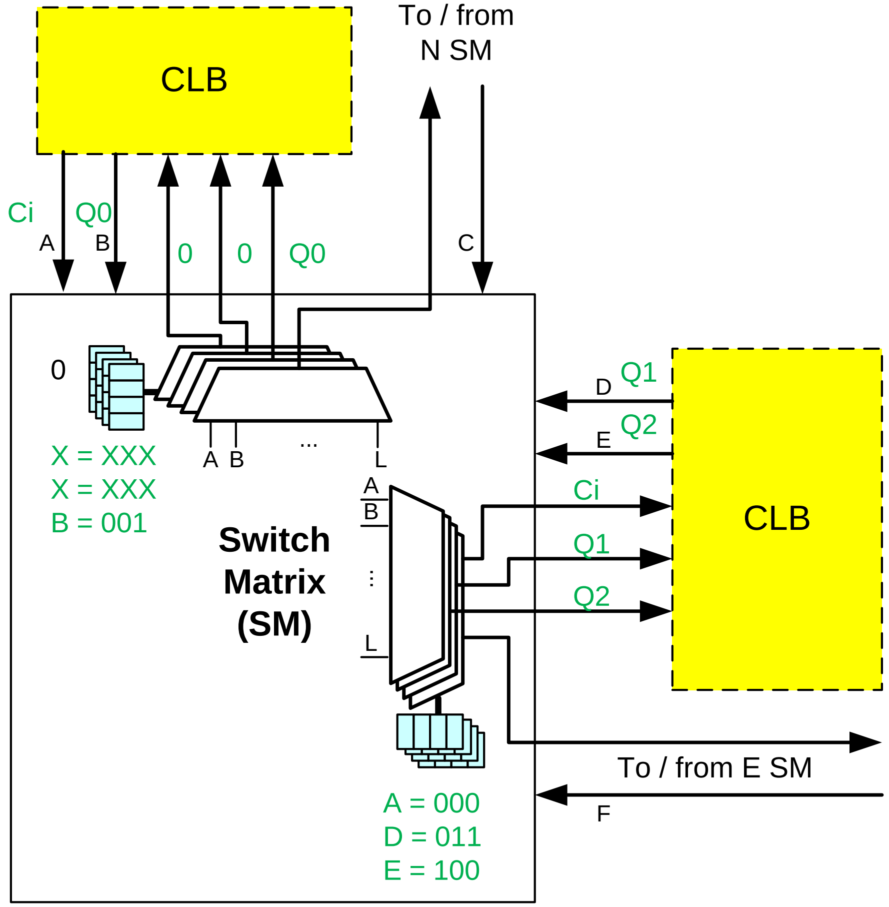


Exercise

- Find the configuration bits to build a 3-bit free-running (always enabled) counter







Implementation

- ASIC's
  - Use the CAD tools to synthesize and route a "netlist"
    - **Synthesis** = Takes logic description or logic schematic & converts to transistor level gates
    - **Place and Route** = Figure out where each gate should go on the chip)
  - Final "netlist" is sent to chip maker for production
  - Fabrication is very expensive (> $1 million) so get your design right the first time.
- FPGA's
  - Synthesis converts logic description to necessary LUT contents, etc.
  - Place and route produces a configuration for the FPGA chip
  - Can reconfigure FPGA as much as you like, so less important to get it right 1st time

ASIC's vs. FPGA's

- ASIC's
  - Faster
  - Handles Larger Designs
  - More Expensive
  - Less Flexible (Cannot be reconfigured to perform a new hardware function)
- FPGA's
  - Slower (extra logic to make it reconfigurable)
  - Smaller Designs
  - Less Expensive
  - Extremely Flexible

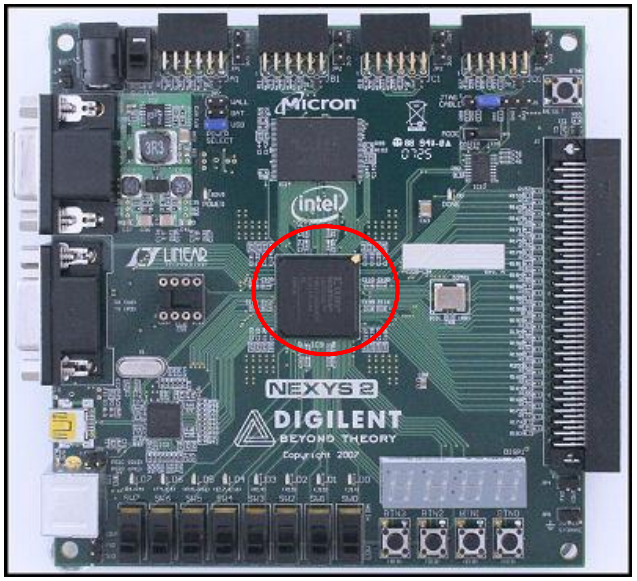
Xilinx Spartan 3E

Digilent Nexys-2 Board

- Has a Xilinx Spartan 3E FPGA (XC3S500e)
- 500K gate equivalent
- 9312 D-FF's on-board

On-board I/O

- (4) 7-Segment Displays
- (8) LED's
- (4) Push Buttons
- (8) Switches



Latest FPGA's

- SoC design (Xilinx Kintex [KU115])
  - Quad-Core ARM cores
  - DDR3 SDRAM Memory Interface
  - ~800 I/O Pins
  - Equiv. ~15M gate equivalent FPGA fabric
    - ~1M D-FFs + 552K LUTs
    - 1968 dedicated DSP "slices" 18x18 multiply + adder
    - 34.6 Megabits of onboard Block RAMs