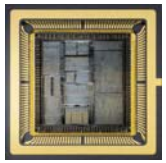
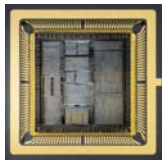


# Lecture 4: Design Flow

## CSCI 5330
## Digital CMOS VLSI Design

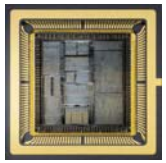
**Instructor**: Saraju P. Mohanty, Ph. D.

**NOTE**: The figures, text etc included in slides are borrowed from various books, websites, authors pages, and other sources for academic purpose only. The instructor does not claim any originality.



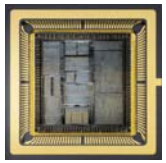
# Lecture Outline

- CMOS Fabrication
- Design Partitioning
- Logic Design
- Circuit Design
- Physical Design
- Fabrication, Packaging, and Testing
- Microwind tool
- DCSH tool



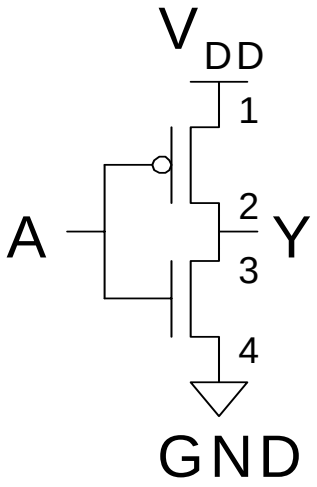
# CMOS Fabrication

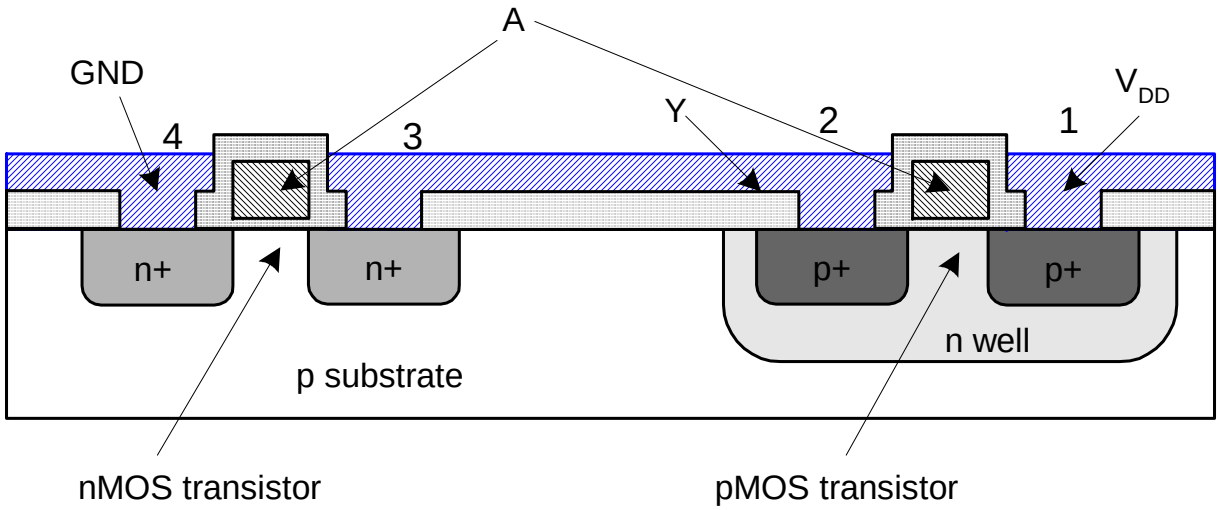
- CMOS transistors are fabricated on silicon wafer.
- Lithography process similar to printing press is used for the fabrication.
- On each step, different materials are deposited or etched.
- Easiest to understand by viewing both top and cross-section of wafer in a simplified manufacturing process.

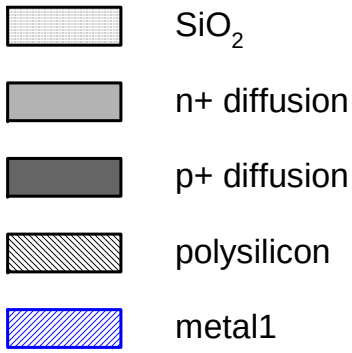


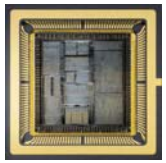
# Inverter Cross-section

- Typically use p-type substrate for nMOS transistors.
- Requires n-well for body of pMOS transistors.



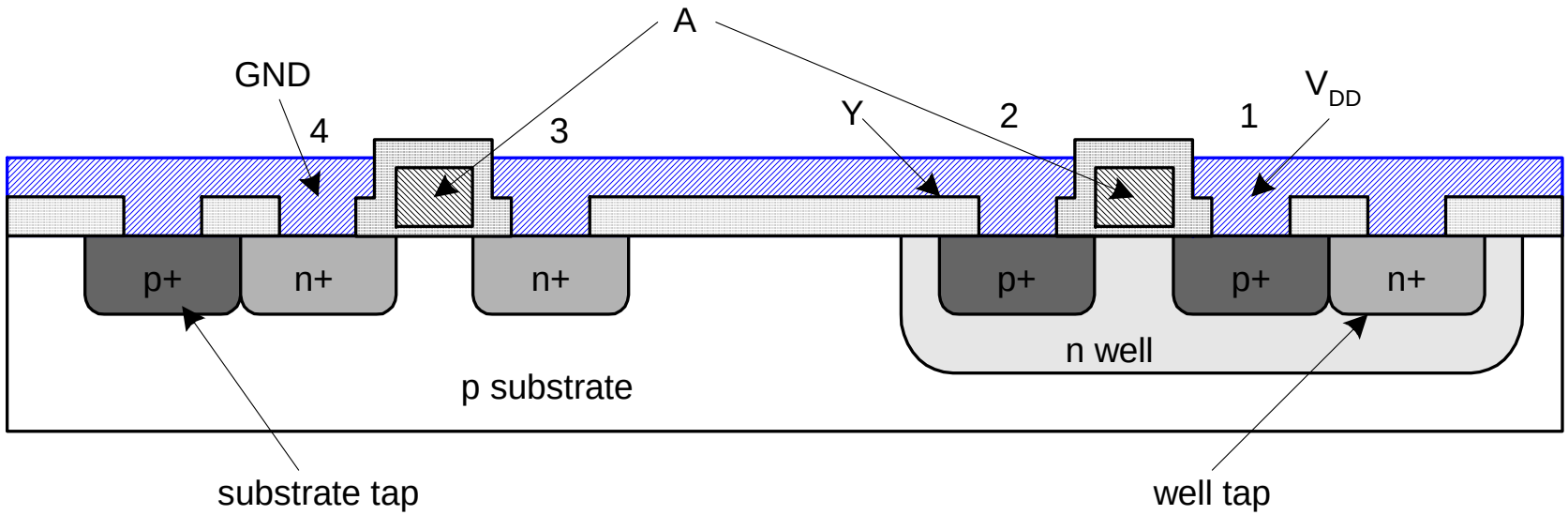



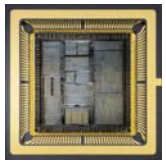





# Well and Substrate Taps

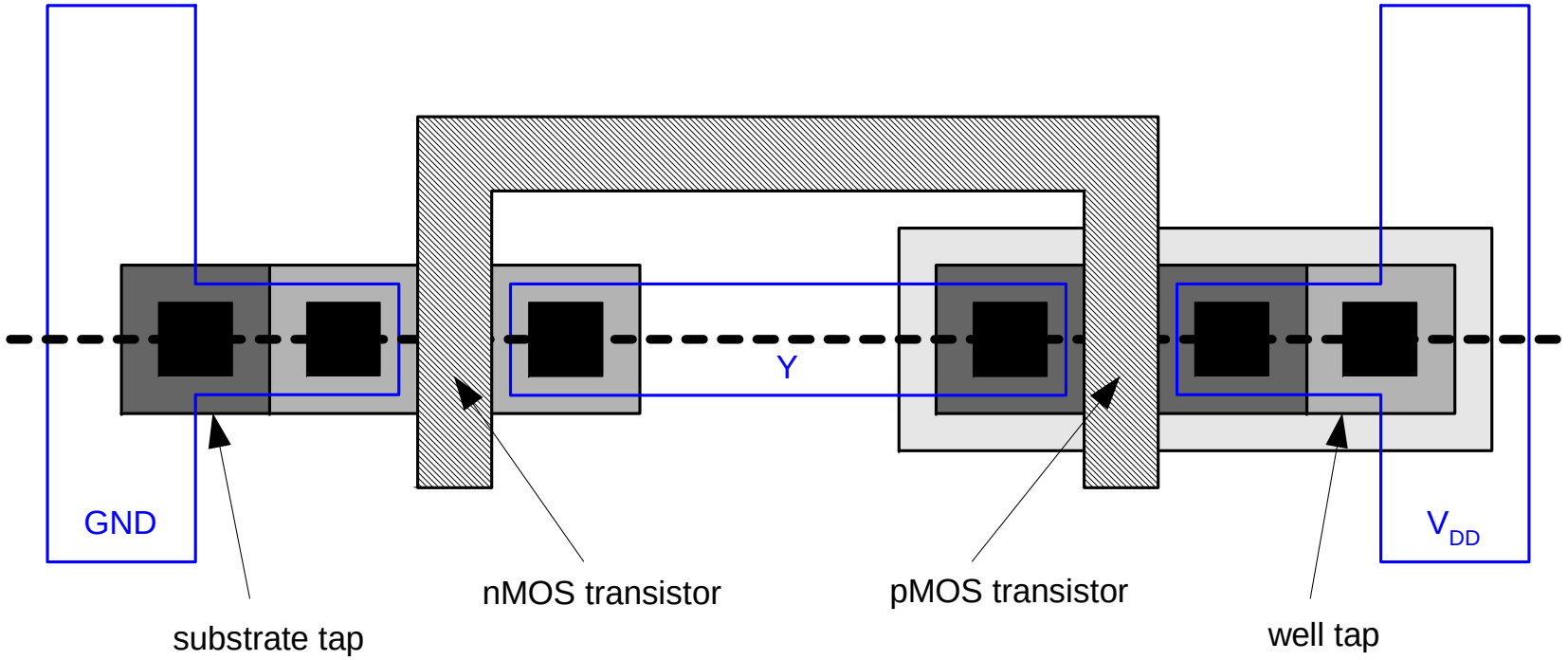
- Substrate must is tied to GND and n-well to $V_{DD}$
- Metal to lightly-doped semiconductor forms poor connection called Shottky Diode
- Heavily doped well and substrate contacts or taps form good ohmic contacts.

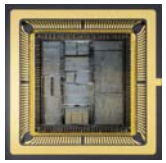




# Inverter Mask Set

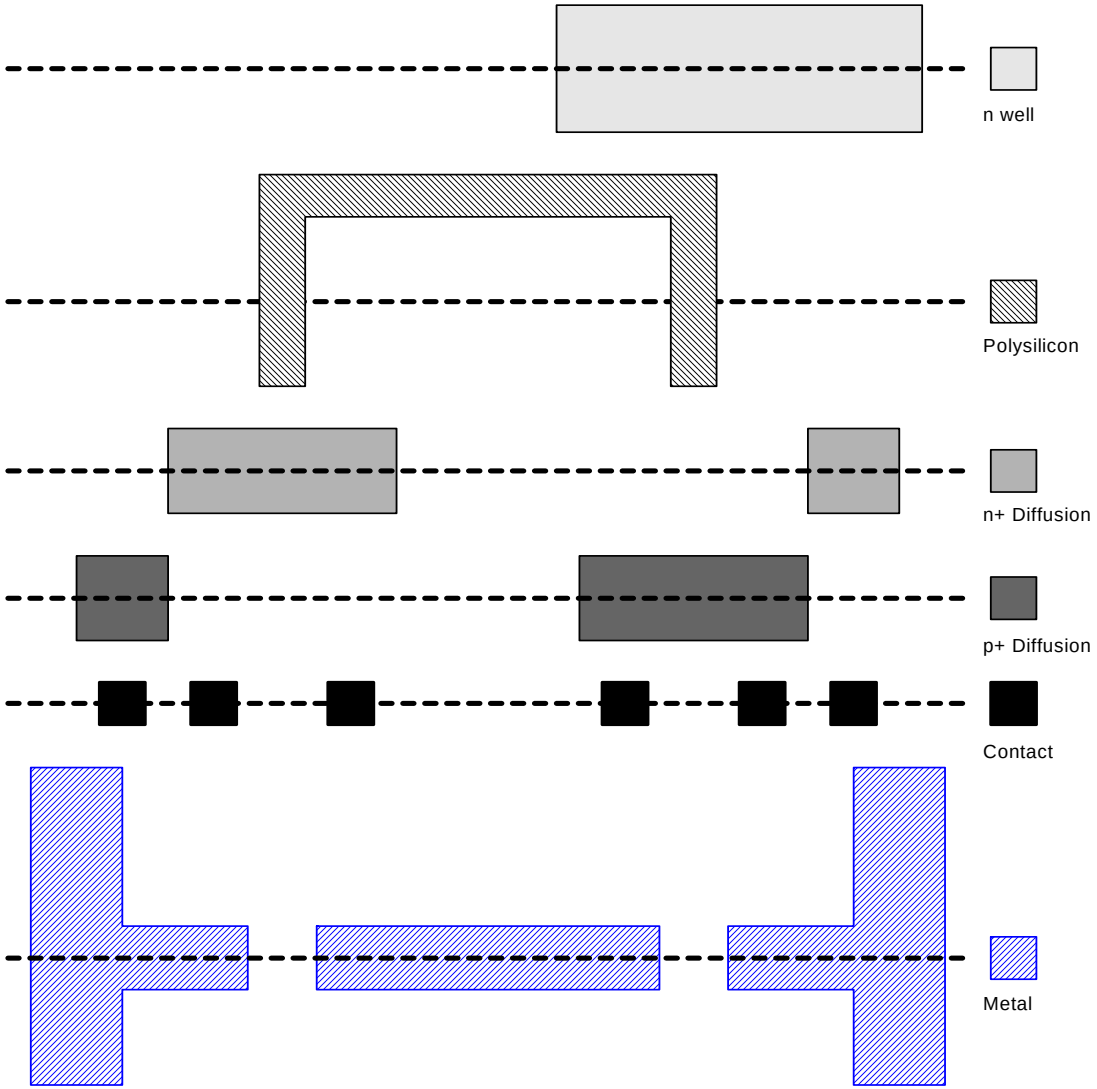
- Transistors and wires are defined by *masks*
- Cross-section taken along dashed line





# Detailed Mask Views

- Six masks
  - n-well
  - Polysilicon
  - n+ diffusion
  - p+ diffusion
  - Contact
  - Metal

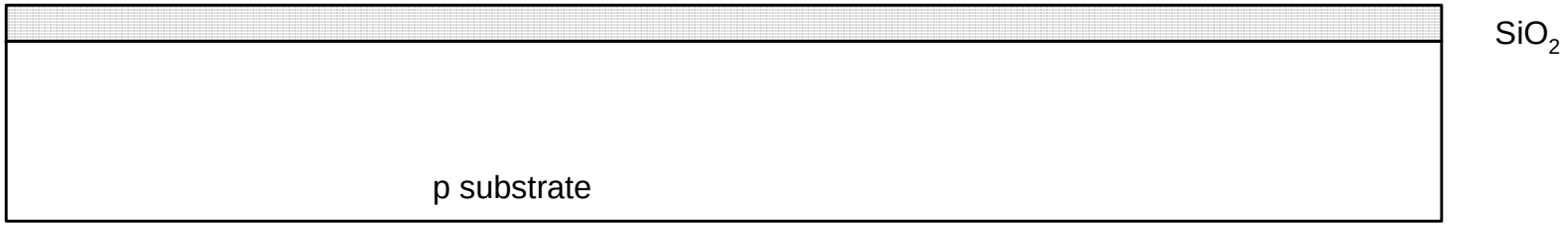


# Fabrication Steps: Creation of n-well

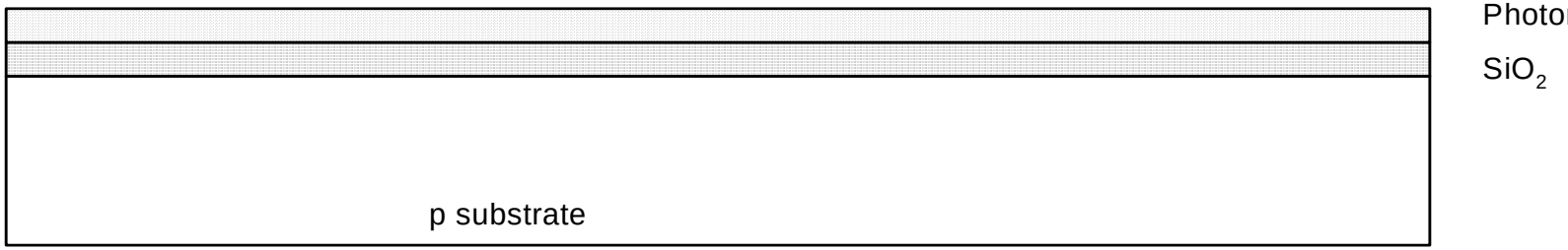
- Objective is to build inverter from the bottom up
- First step will be to form the n-well
  - Cover wafer with protective layer of $SiO_2$ (oxide)
  - Remove layer where n-well should be built
  - Implant or diffuse n dopants into exposed wafer
  - Strip off $SiO_2$
- n-well : Start with blank p-type silicon wafer

# Fabrication Steps: Creation of n-well

- n-well: Grow $SiO_2$ on top of Si wafer
  - 900 – 1200 C with $H_2O$ or $O_2$ in oxidation furnace
  - The oxide is patterned to define n-well.

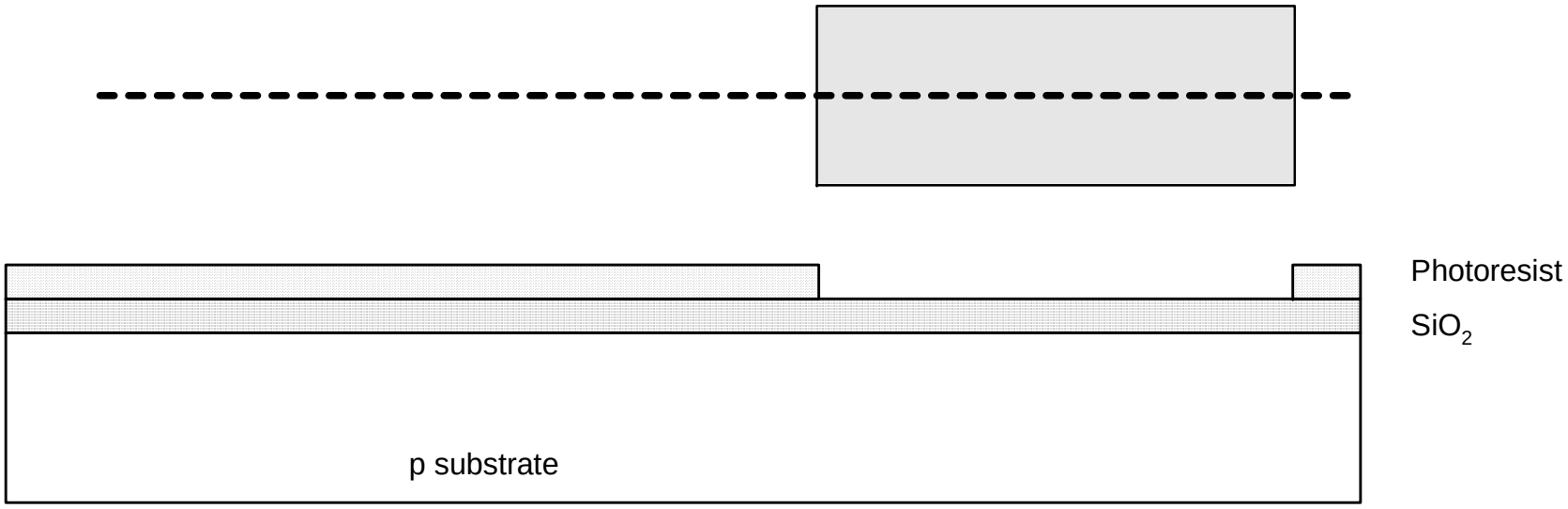



- n-well: Spin on photoresist
  - Photoresist is a light-sensitive organic polymer
  - Softens where exposed to light



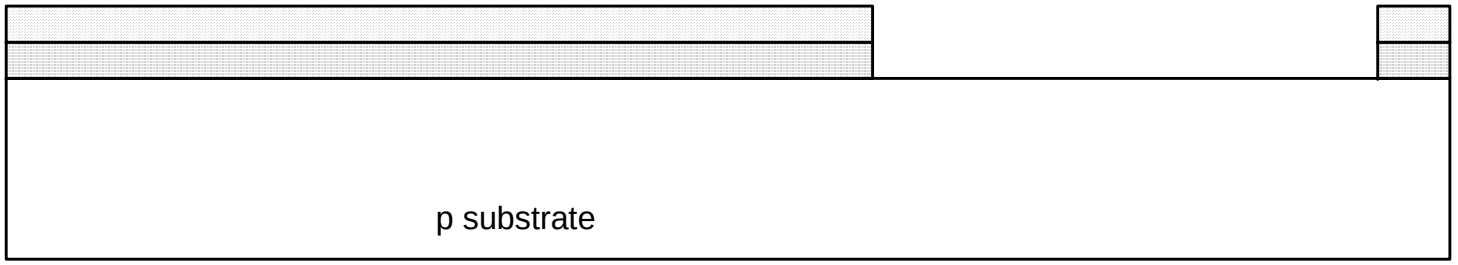
# Fabrication Steps: Creation of n-well

- n-well: Expose photoresist through n-well mask
  - Allows light to pass through only where the n-well need to be created.
  - Strip off exposed photoresist

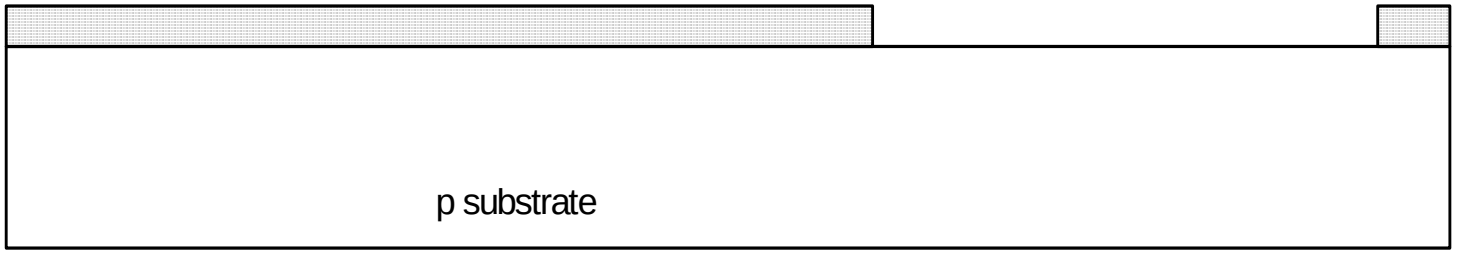


# Fabrication Steps: Creation of n-well

- n-well: Etch oxide with hydrofluoric acid (HF)
  - Only attacks oxide where resist has been exposed

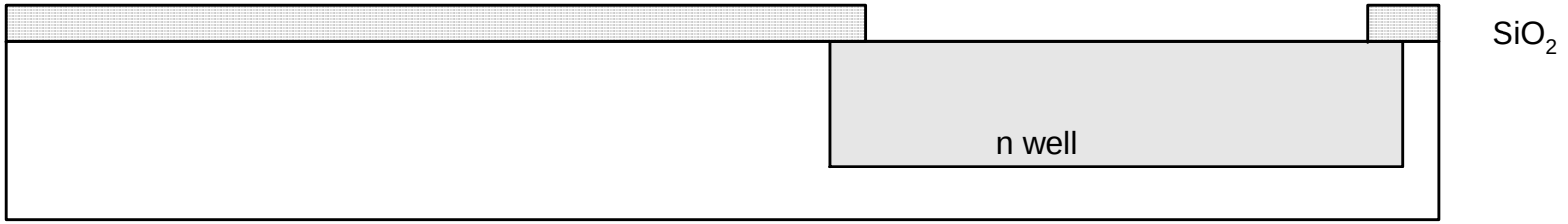



- n-well: Strip off remaining photoresist
  - Use mixture of acids called piranah etch
  - Necessary so resist doesn't melt in next step

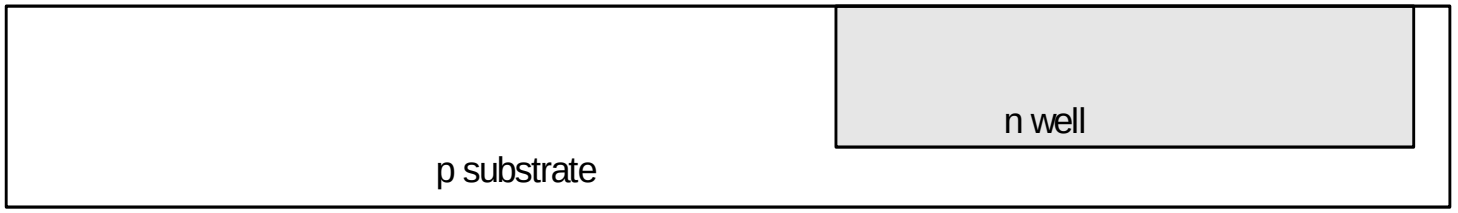


# Fabrication Steps: Creation of n-well

- n-well: created with diffusion or ion implantation
  - Diffusion: Place wafer in furnace with arsenic gas and heat until As atoms diffuse into exposed Si
  - Ion Implantation: Blast wafer with beam of As ions

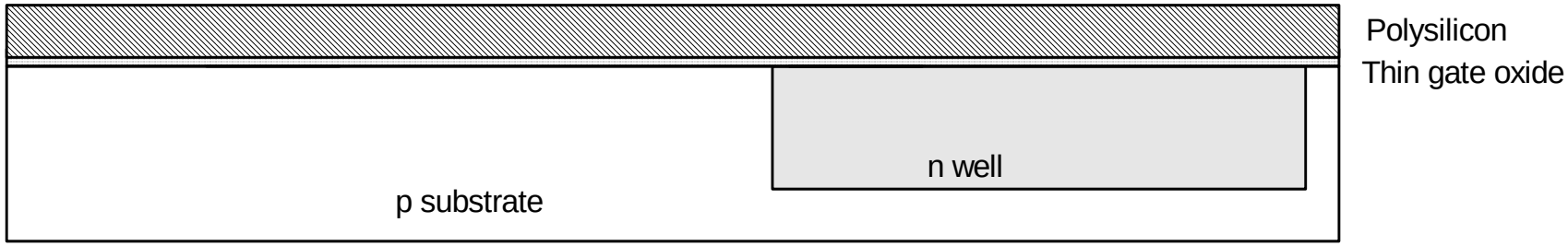



- n-well: Strip off the remaining oxide using HF
  - Back to bare wafer with n-well
  - Subsequent steps involve similar series of steps



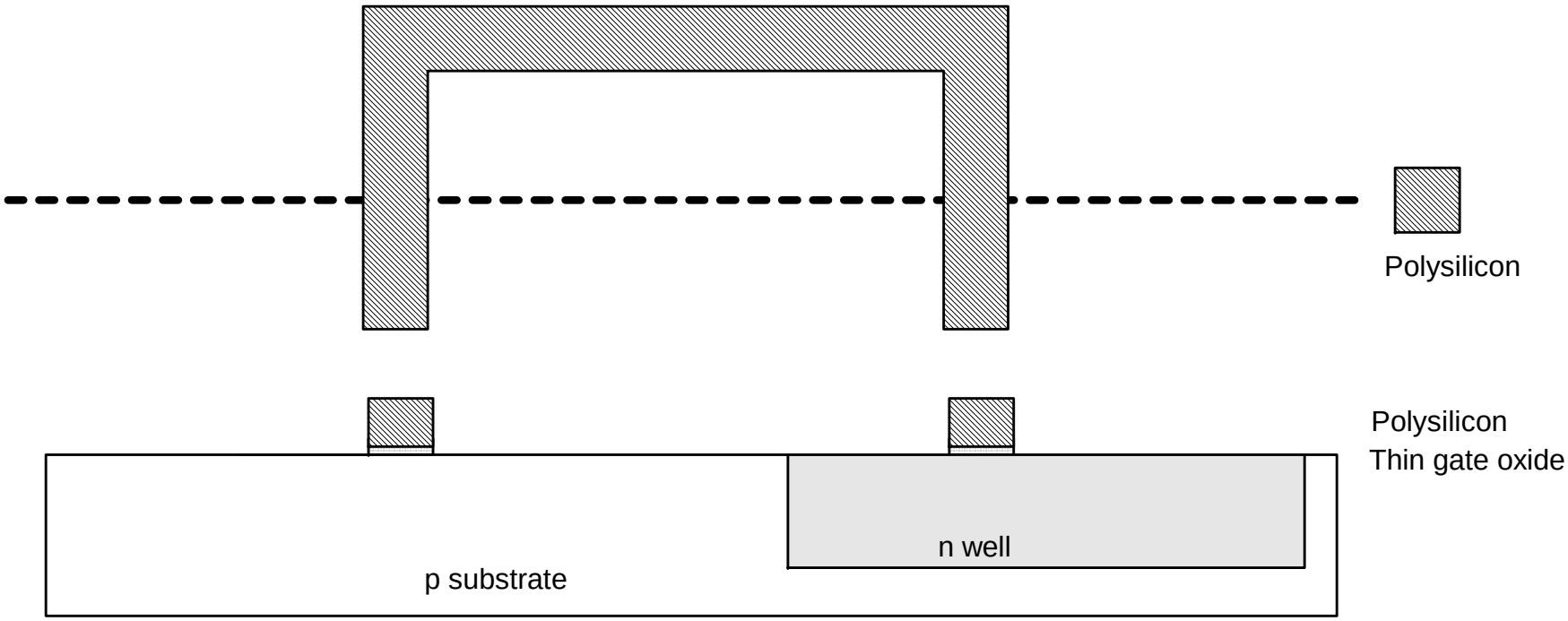
# Fabrication Steps: Creation of Gates

- Gate consists of polysilicon over thin layer of silicon oxide.
- Very thin layer of gate oxide is grown in furnace
  - < 20 Å (6-7 atomic layers)
- Chemical Vapor Deposition (CVD) of silicon layer for polysilicon deposition
  - Place wafer in furnace with Silane gas ($SiH_4$)
  - Forms many small crystals called polysilicon
  - Polysilicon is heavily doped to be a good conductor



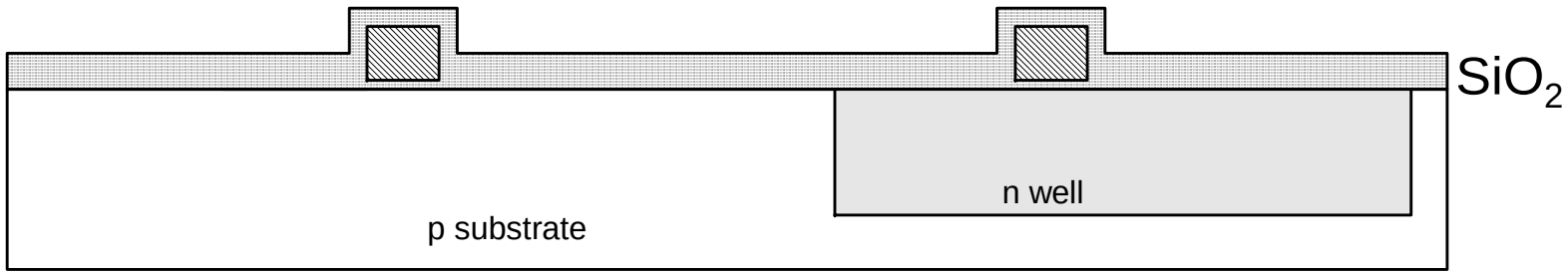
# Fabrication Steps: Creation of Gates

- Use same lithography process that used to create n-well to pattern polysilicon using photoresist and the polysilicon mask.



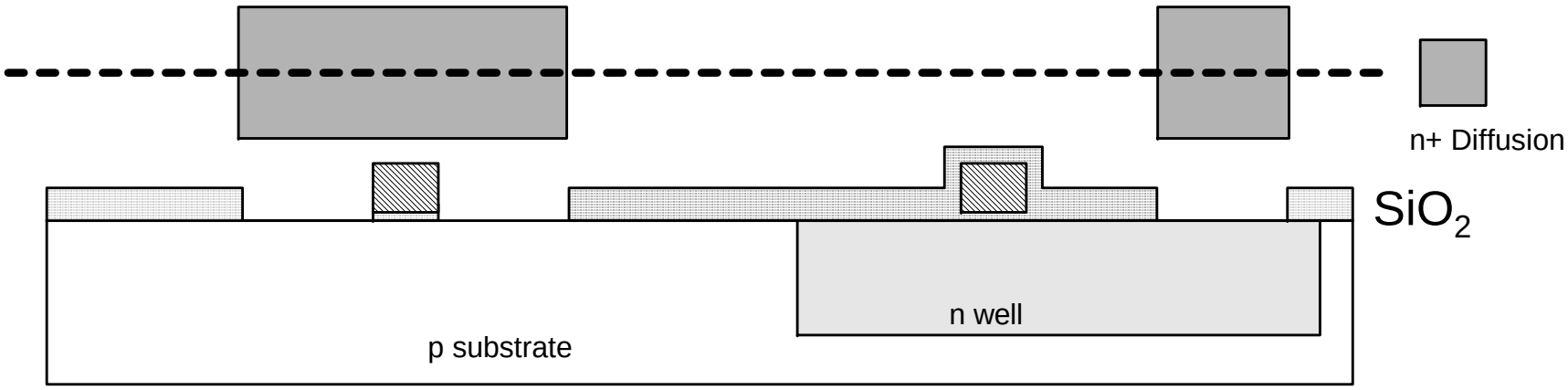
# Fabrication Steps: Creation of n+

- Transistor active area and well contact are n+.
- N-diffusion forms nMOS source, drain, and n-well contact
- Use oxide and masking to expose where n+ dopants should be diffused or implanted



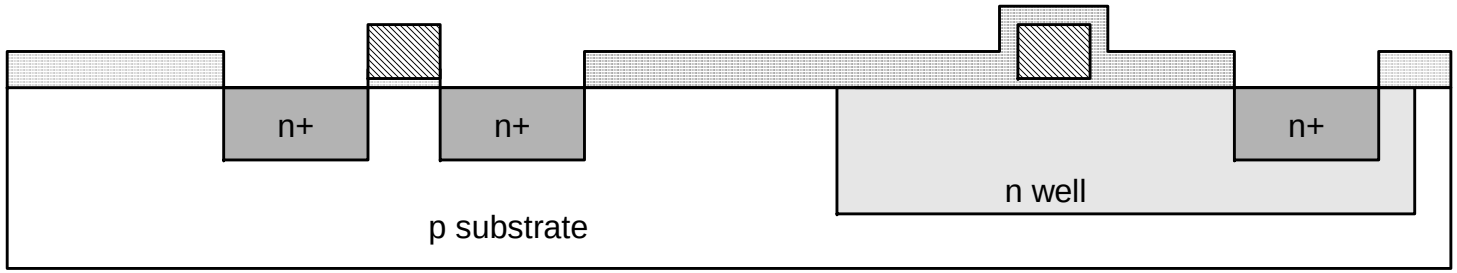
# Fabrication Steps: Creation of n+

- Pattern oxide with the n-diffusion mask and form n+ regions
- *Self-aligned process* where gate blocks diffusion
- Polysilicon is better than metal for self-aligned gates because it doesn't melt during later processing

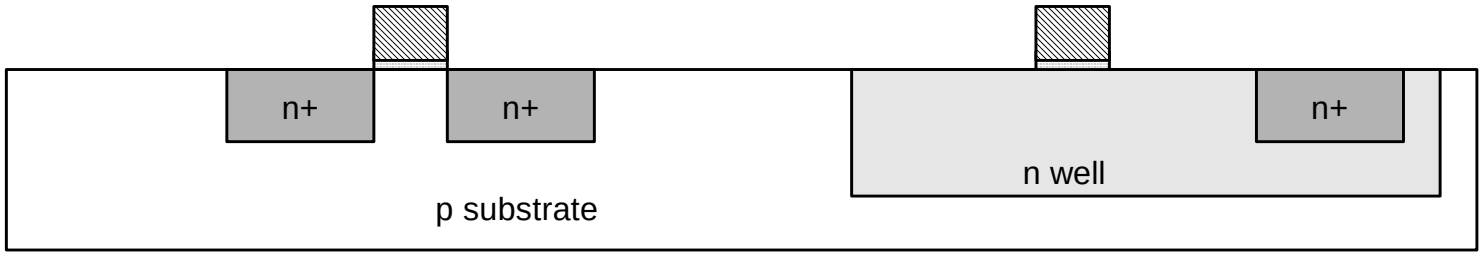


# Fabrication Steps: Creation of n+

- Historically dopants were diffused
- Usually ion implantation today
- But regions are still called diffusion

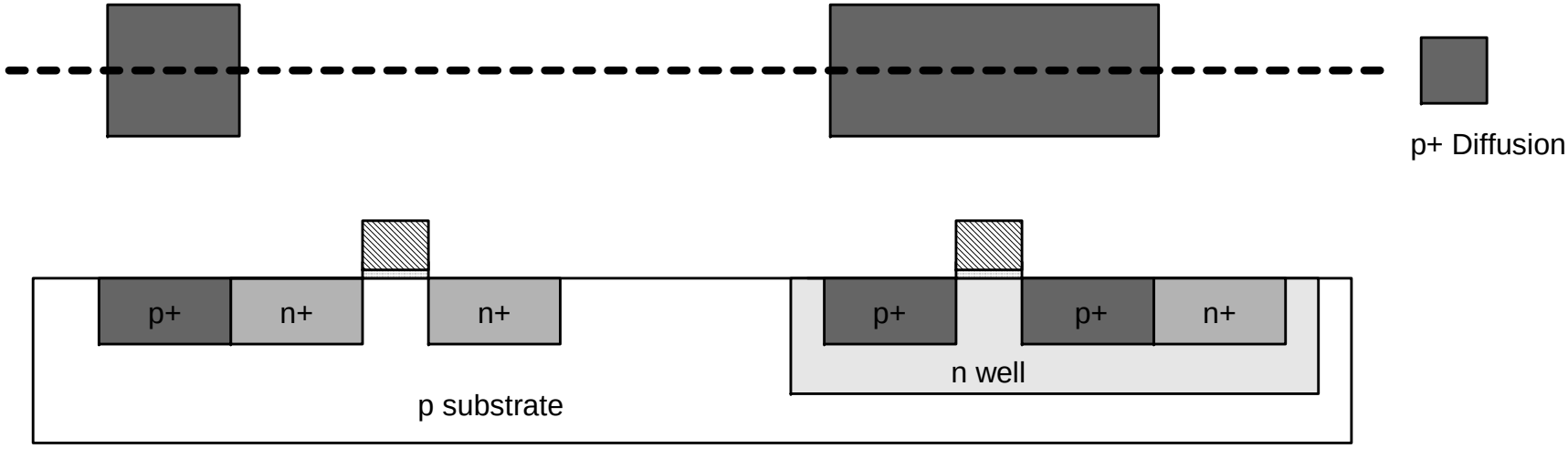



- Strip off oxide to complete patterning step



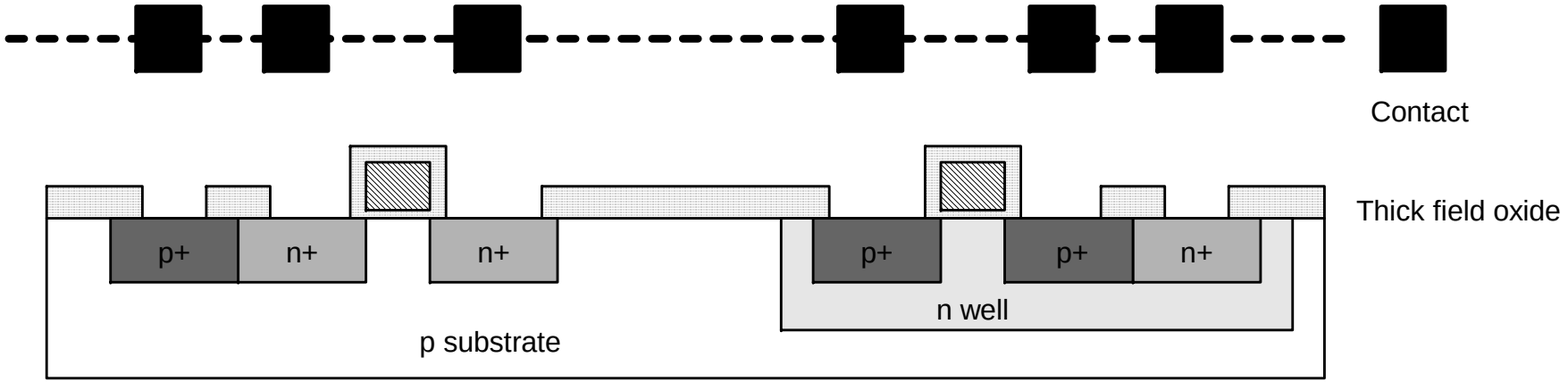
# Fabrication Steps: Creation of p+

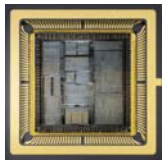
- Similar set of steps form p+ diffusion regions for pMOS source and drain and substrate contact
- Pattern oxide with the p-diffusion mask and form p+ regions



# Fabrication Steps: Creation of Contacts

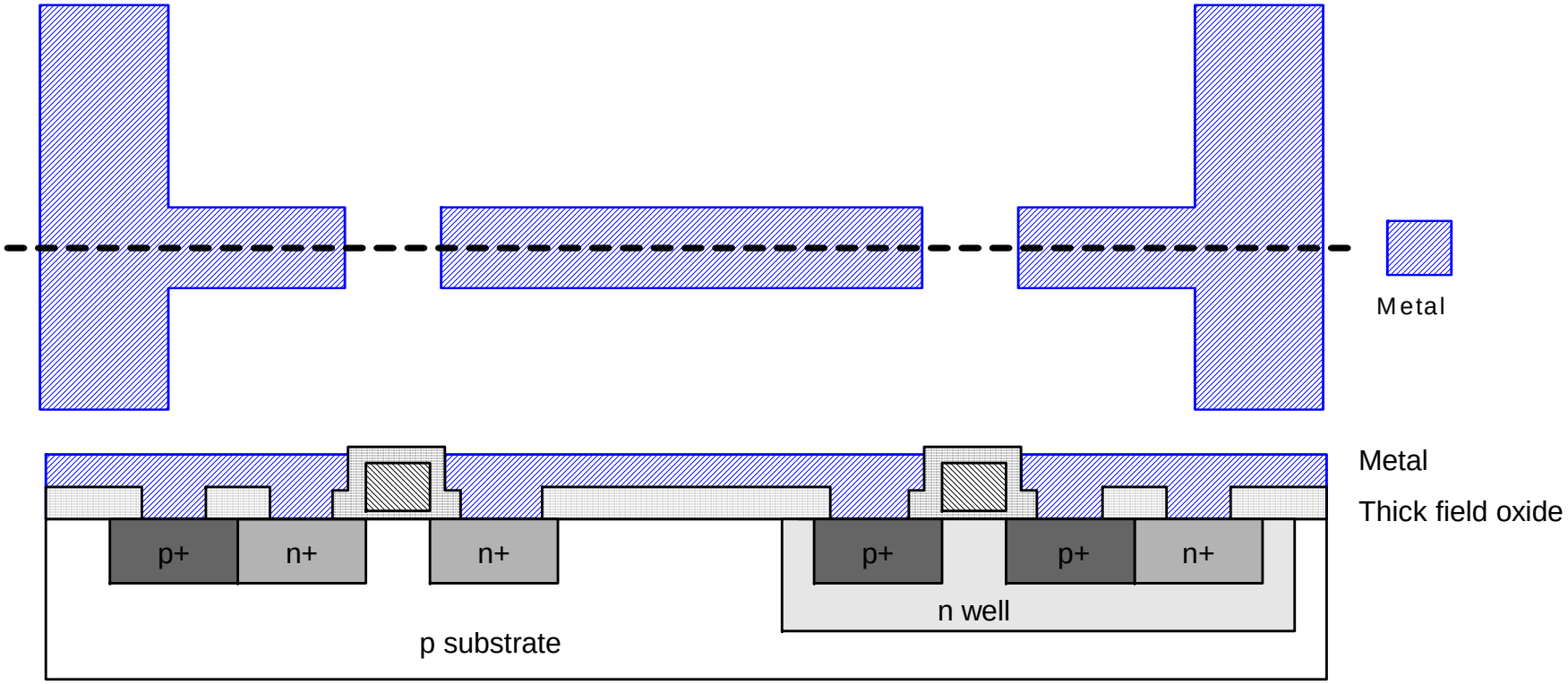
- Now we need to wire together the devices
- Cover chip with thick field oxide
- Etch oxide where contact cuts are needed using contact mask.

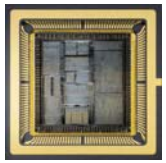




# Fabrication Steps: Metalization

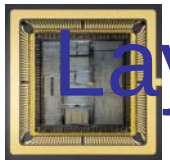
- Sputter on aluminum over whole wafer
- Pattern to remove excess metal, leaving wires
- Metal mask is used during this step.





# Layout Design Rules

- Chips are specified with set of masks
- Minimum dimensions of masks determine transistor size (and hence speed, cost, and power)
- Feature size *f* = distance between source and drain
  - Set by minimum width of polysilicon
- Normalize for feature size when describing design rules
- Express rules in terms of $\lambda = f/2$
  - e.g. $\lambda$ = 60 nm in 120 nm process



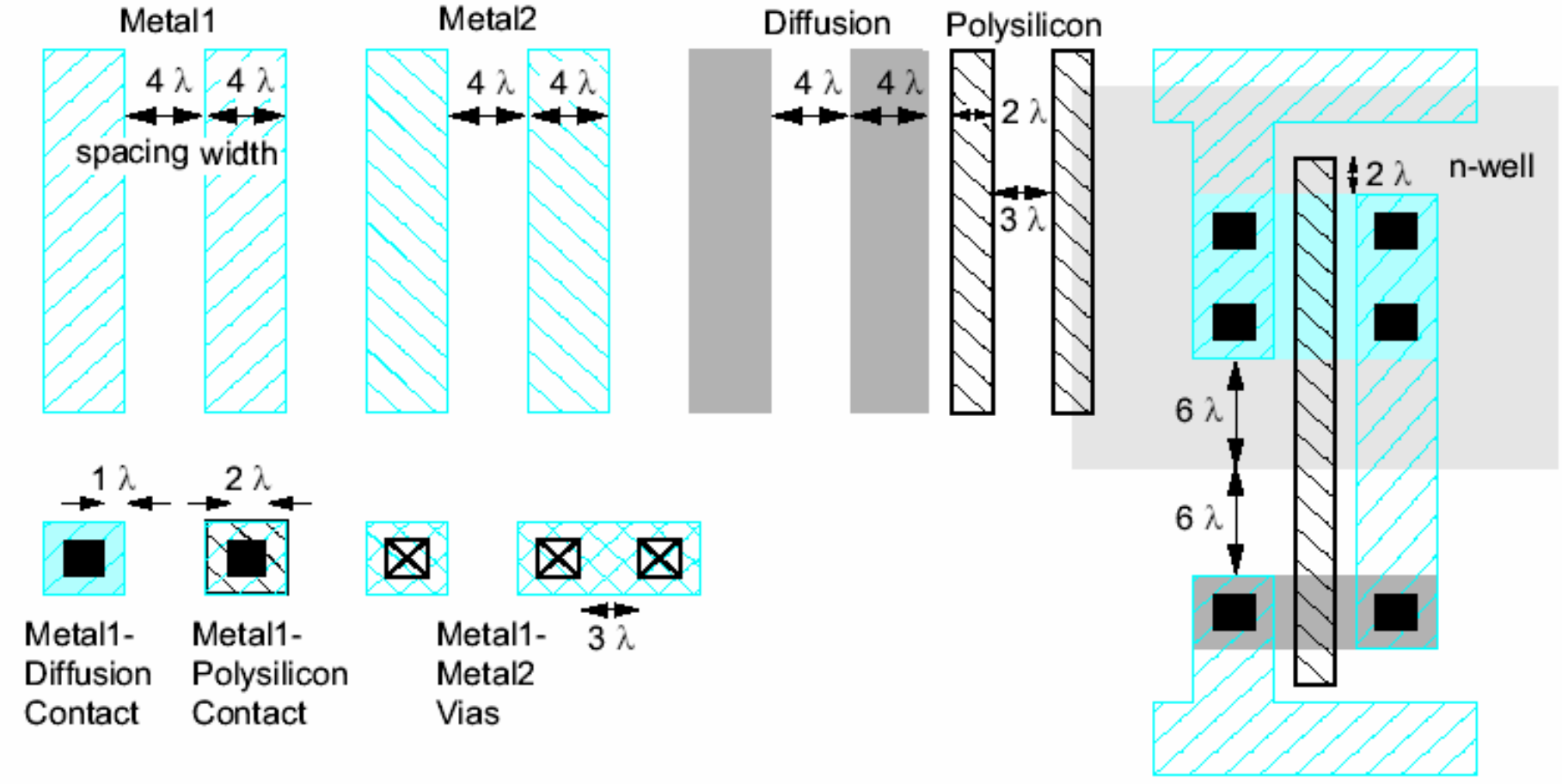
# Layout Design Rules : Simplified Form

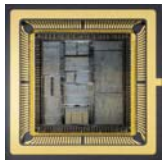
- Metal and diffusion have minimum width and spacing of 4λ.
- Contacts are 2λ x 2λ and surrounded by 1λ layer.
- Polysilicon width is 2λ.
- Polysilicon overlaps diffusion by 2λ or 1λ depending on situation.
- Polysilicon and contacts have spacing 3λ.
- n-well surrounds PMOS transistors by 6λ and NMOS transistors 6λ.

NOTE: Do not have to remember, CAD softwares handle it automatically once DRC are set depending on the foundry chosen.

# Layout Design Rules : Simplified Form

- Conservative rules to get you started

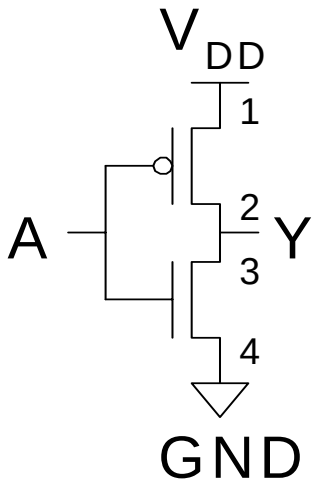


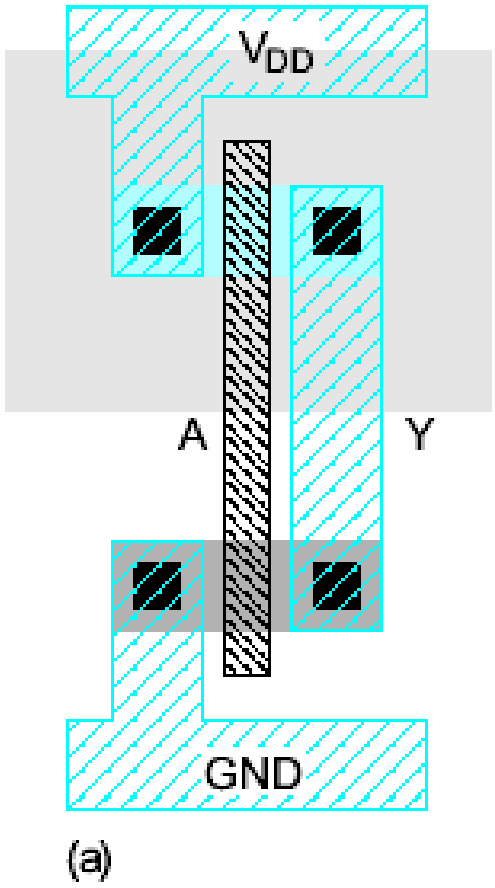


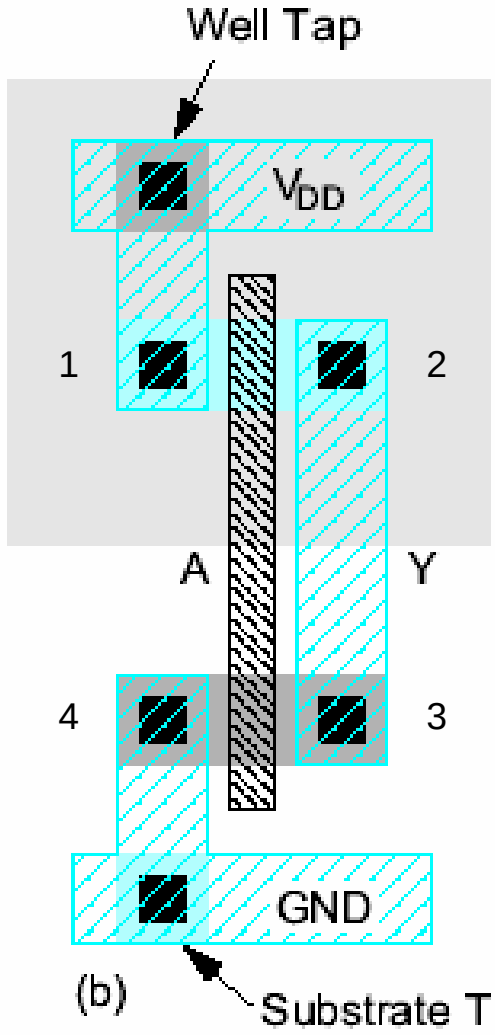
# Logic Gate Layout

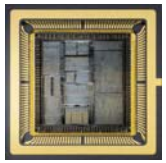
- Layout can be very time consuming
  - Design gates to fit together nicely
  - Build a library of standard cells
- Standard cell design methodology
  - $V_{DD}$ and GND should abut (standard height)
  - Adjacent gates should satisfy design rules
  - nMOS at bottom and pMOS at top
  - All gates include well and substrate contacts

# Logic Gate Layout : Inverter



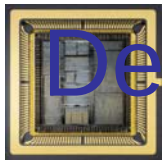





# Design Partitioning

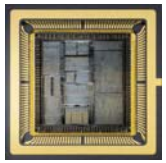
- How to design modern System-on-Chip (SOC)?
  - Many millions (soon billions!) of transistors
  - Tens to hundreds of engineers

- Approaches for larger system design:
  - Structured Design: Uses principles of Hierarchy, Regularity, Modularity, and Locality
  - Design Partitioning: Design is partitioned to five interrelated tasks, such as architecture design, microarchitecture design, logic design, circuit design and physical design.

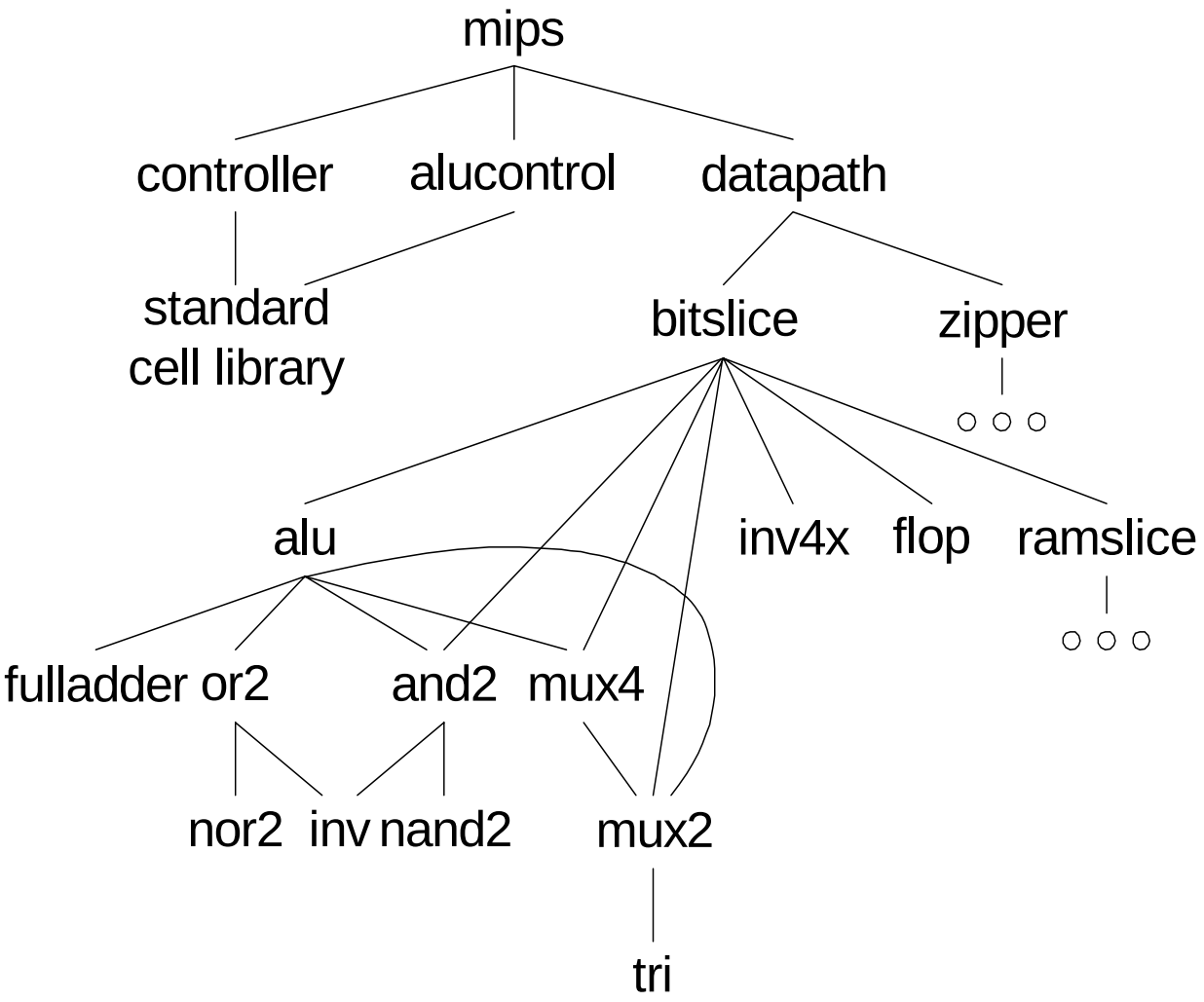


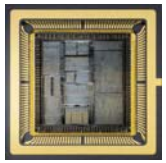
# Design Partitioning : Structured Design

- **Hierarchy**: Divide and Conquer
  - Recursively system into modules
- **Regularity**
  - Reuse modules wherever possible
  - Ex: Standard cell library
- **Modularity**: well-formed interfaces
  - Allows modules to be treated as black boxes
- **Locality**
  - Physical and temporal



# Hierarchical Design

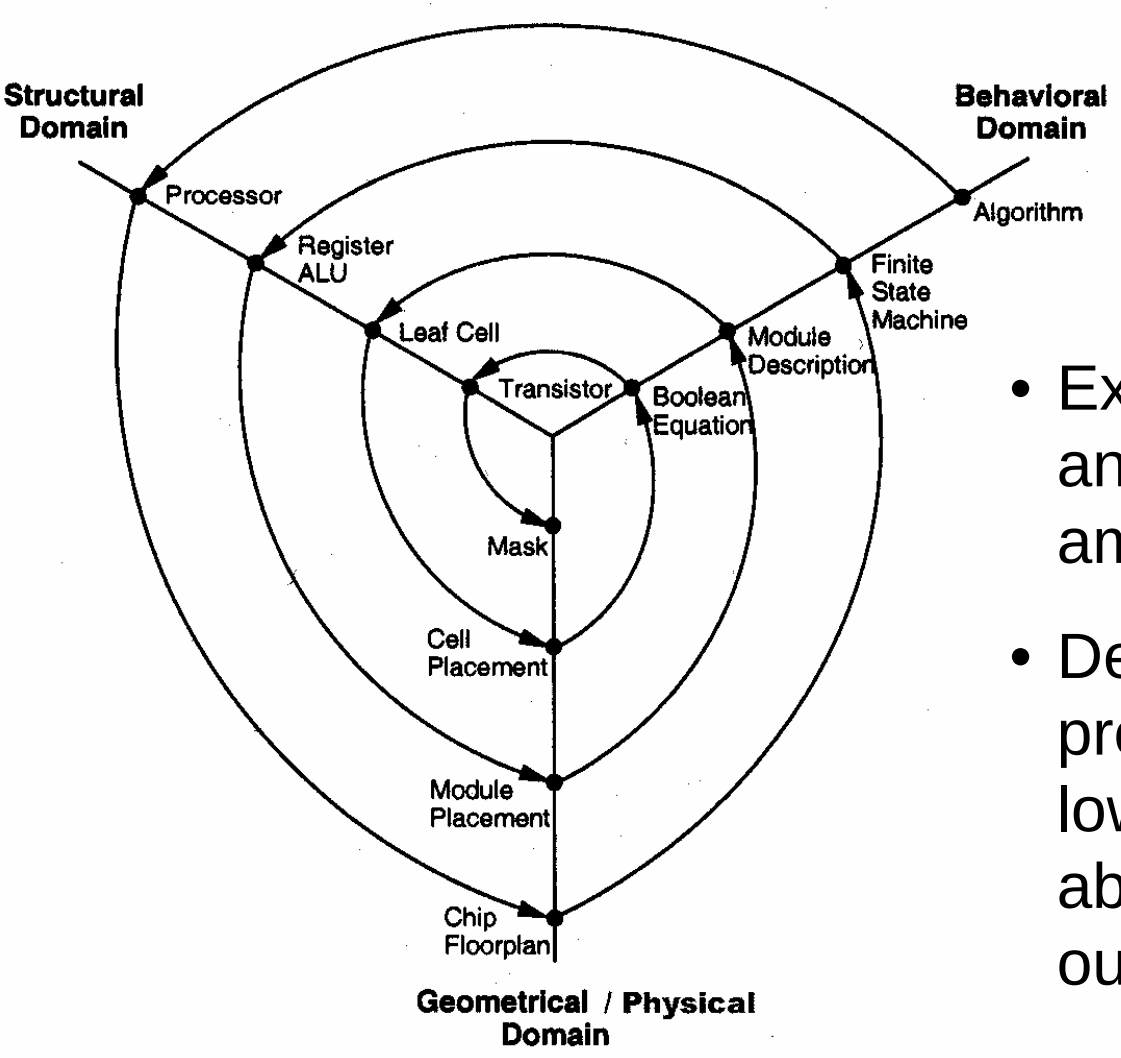




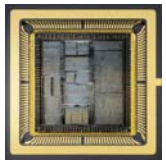
# Design Partitioning : Y-Chart

- **Architecture**: User's perspective, what does it do?
  - Instruction set, registers
  - MIPS, x86, Alpha, PIC, ARM, …
- **Microarchitecture:**
  - Single cycle, multicycle, pipelined, superscalar?
- **Logic**: how are functional blocks constructed
  - Ripple carry, carry look-ahead, carry select adders
- **Circuit**: how are transistors used
  - Complementary CMOS, pass transistors, domino
- **Physical**: chip layout
  - Datapaths, memories, random logic

# Design Partitioning : Y-Chart



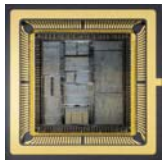

- Explains each domain and transformation among domains.
- Design process proceeds from higher to lower levels of abstractions i.e. from outer to inner rings.



# HDLs

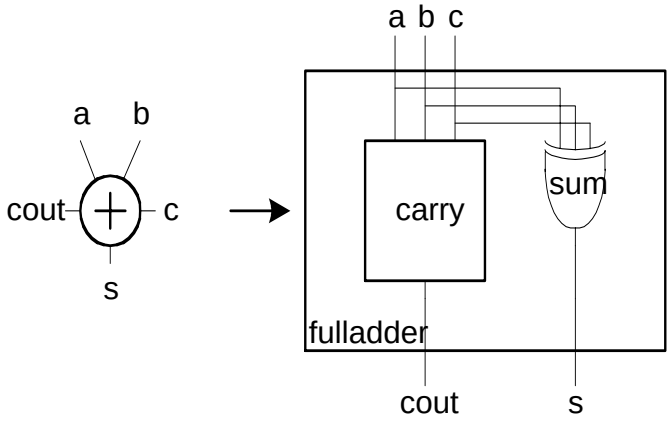
- Hardware Description Languages
  - Widely used in logic design
  - Verilog and VHDL

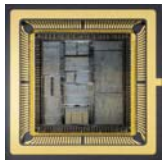
- Describe hardware using code
  - Document logic functions
  - Simulate logic before building
  - Synthesize code into gates and layout
    - Requires a library of standard cells



# HDLs: Verilog Example

```
module fulladder
  (input a, b, c, output s, cout);
sum  s1(a, b, c, s);
carry c1(a, b, c, cout);
endmodule
module carry (input a, b, c,
              output cout)
assign cout
          = (a&b) | (a&c) | (b&c);
endmodule
```



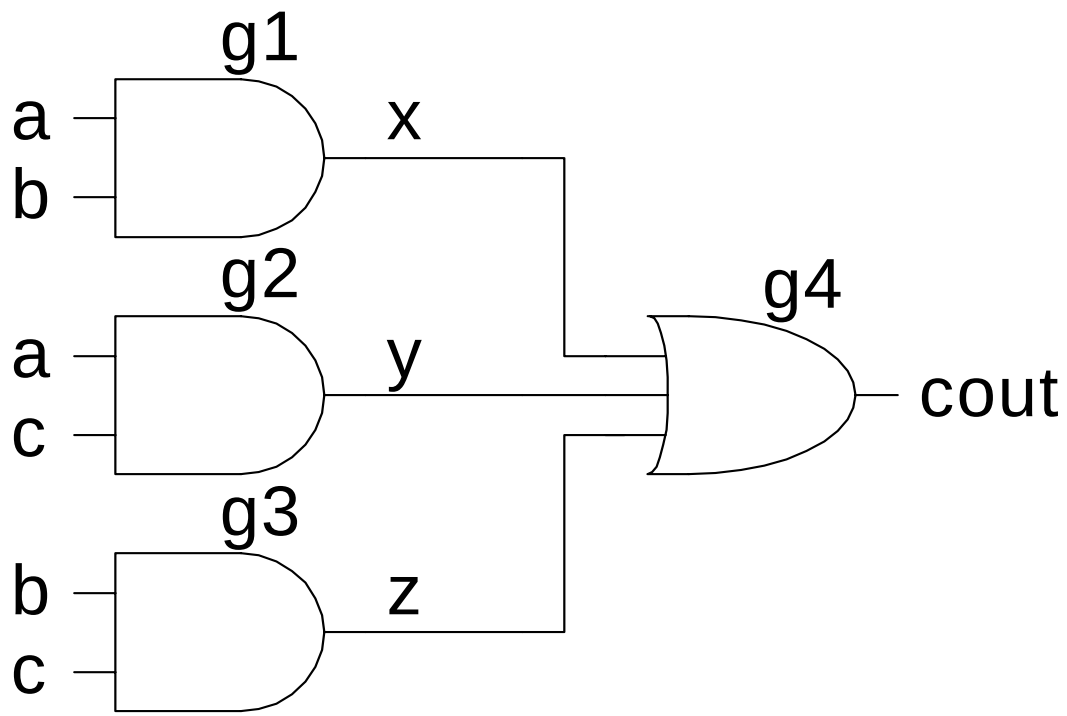


# Circuit Design

- How should logic be implemented?
  - NANDs and NORs vs. ANDs and ORs?
  - Fan-in and fan-out?
  - How wide should transistors be?
- These choices affect speed, area, power
- Logic synthesis makes these choices for you
  - Good enough for many applications
  - Hand-crafted circuits are still better

# Carry Logic Example: Logic Level

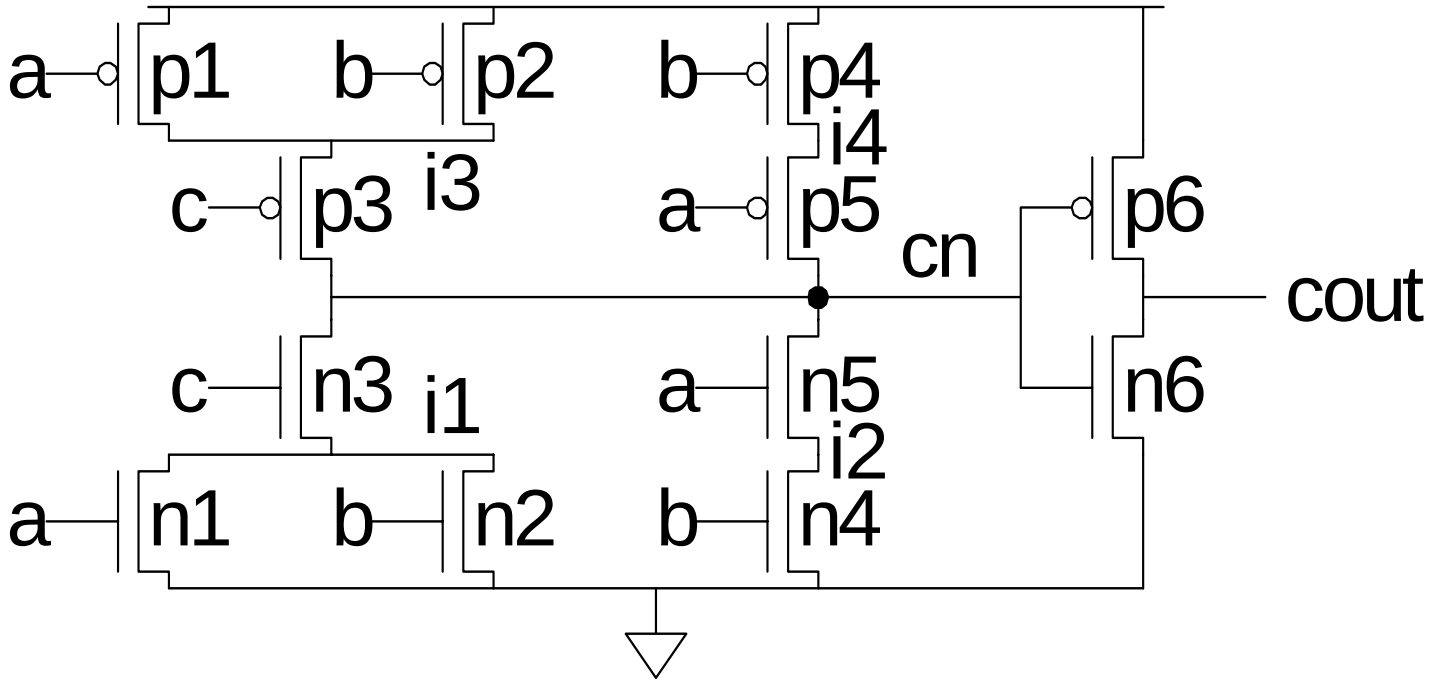
- **assign** cout = (a&b) | (a&c) | (b&c);




Transistors?  Gate Delays?

# Carry Logic Example: Transistor Level

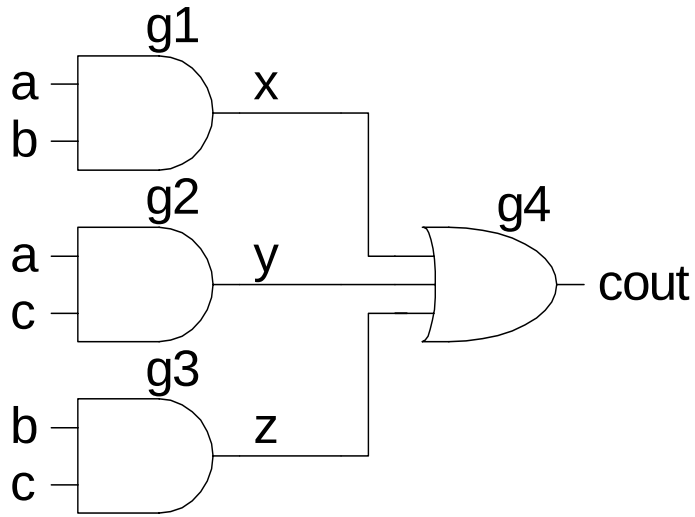
- **assign** cout = (a&b) | (a&c) | (b&c);




Transistors? Gate Delays?

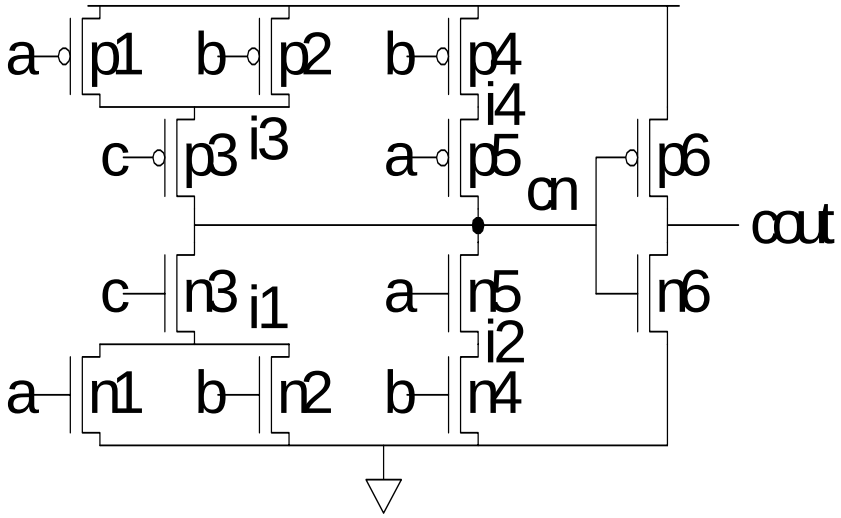
# Carry Logic Example: Gate-level Netlist

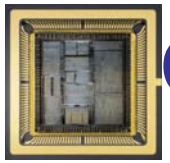
```
module carry
   (input  a, b, c, output cout)

        wire   x, y, z;
        and g1(x, a, b);
        and g2(y, a, c);
        and g3(z, b, c);
        or  g4(cout, x, y, z);

endmodule
```



# Carry Logic Example: Transistor-Level Netlist

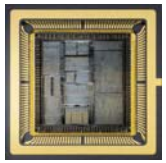
```
module carry
   (input  a, b, c, output cout)
          wire      i1, i2, i3, i4, cn;
          tranif1 n1(i1, 0, a);
          tranif1 n2(i1, 0, b);
          tranif1 n3(cn, i1, c);
          tranif1 n4(i2, 0, b);
          tranif1 n5(cn, i2, a);
          tranif0 p1(i3, 1, a);
          tranif0 p2(i3, 1, b);
          tranif0 p3(cn, i3, c);
          tranif0 p4(i4, 1, b);
          tranif0 p5(cn, i4, a);
          tranif1 n6(cout, 0, cn);
          tranif0 p6(cout, 1, cn);
endmodule
```





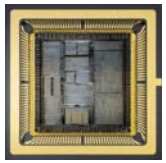
# Carry Logic Example: SPICE Netlist

```
.SUBCKT CARRY A B C COUT VDD GND
MN1 I1 A GND GND NMOS W=1U L=0.18U AD=0.3P AS=0.5P
MN2 I1 B GND GND NMOS W=1U L=0.18U AD=0.3P AS=0.5P
MN3 CN C I1 GND NMOS W=1U L=0.18U AD=0.5P AS=0.5P
MN4 I2 B GND GND NMOS W=1U L=0.18U AD=0.15P AS=0.5P
MN5 CN A I2 GND NMOS W=1U L=0.18U AD=0.5P AS=0.15P
MP1 I3 A VDD VDD PMOS W=2U L=0.18U AD=0.6P AS=1 P
MP2 I3 B VDD VDD PMOS W=2U L=0.18U AD=0.6P AS=1P
MP3 CN C I3 VDD PMOS W=2U L=0.18U AD=1P AS=1P
MP4 I4 B VDD VDD PMOS W=2U L=0.18U AD=0.3P AS=1P
MP5 CN A I4 VDD PMOS W=2U L=0.18U AD=1P AS=0.3P
MN6 COUT CN GND GND NMOS W=2U L=0.18U AD=1P AS=1P
MP6 COUT CN VDD VDD PMOS W=4U L=0.18U AD=2P AS=2P
CI1 I1 GND 2FF
CI3 I3 GND 3FF
CA A GND 4FF
CB B GND 4FF
CC C GND 2FF
CCN CN GND 4FF
CCOUT COUT GND 2FF
.ENDS
```

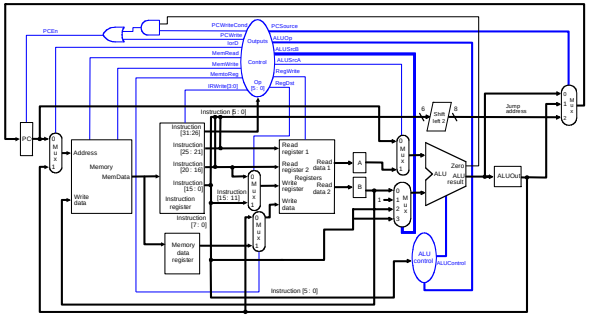


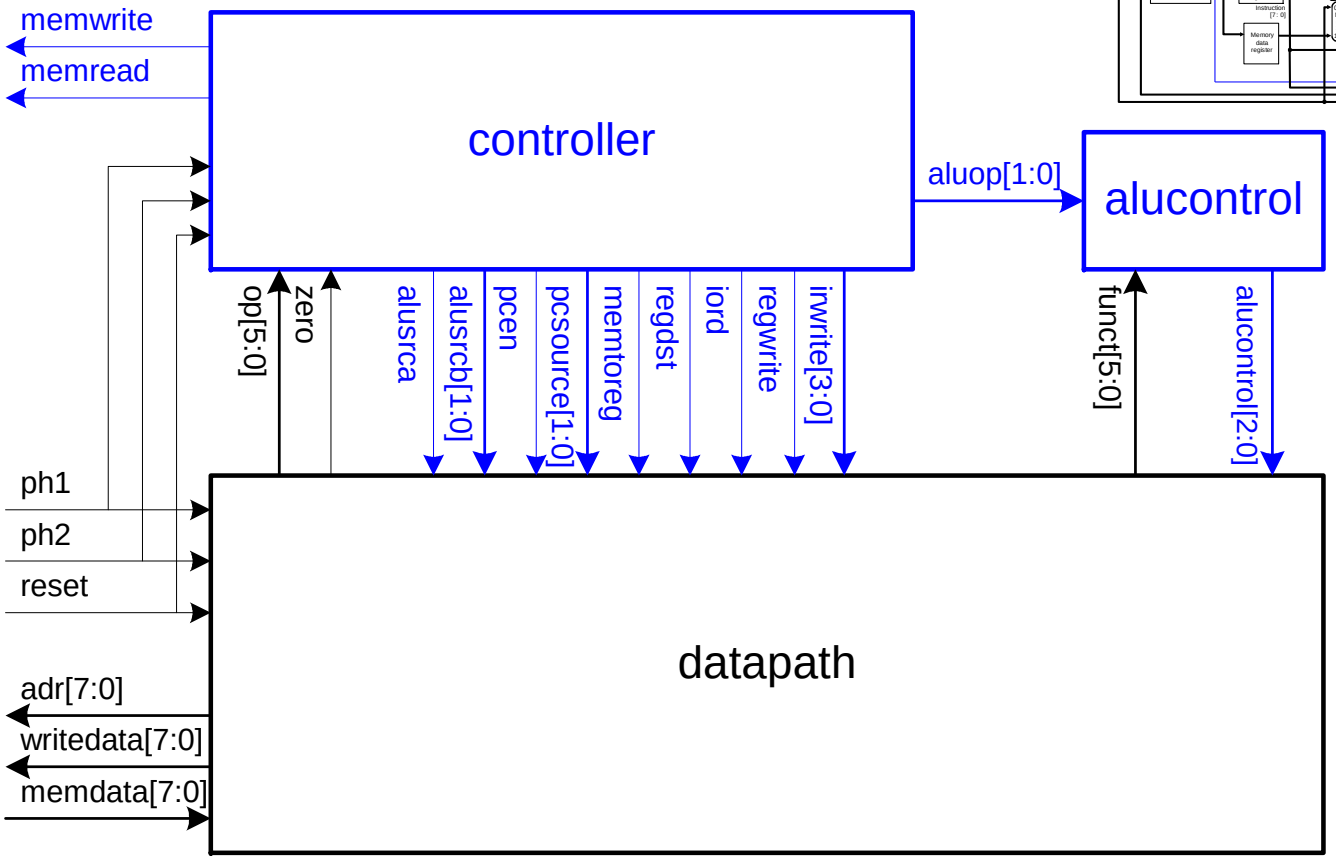
# Physical Design

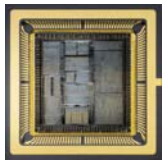
- Floorplan : First step. Determines if design will fit in are budget.
- Standard cells : Layout is often generated using automatic place and route.
- Slice planning : Divide to slices. Slice plan makes it is easy to calculate wire length, estimate area, and evaluate wire congestion.



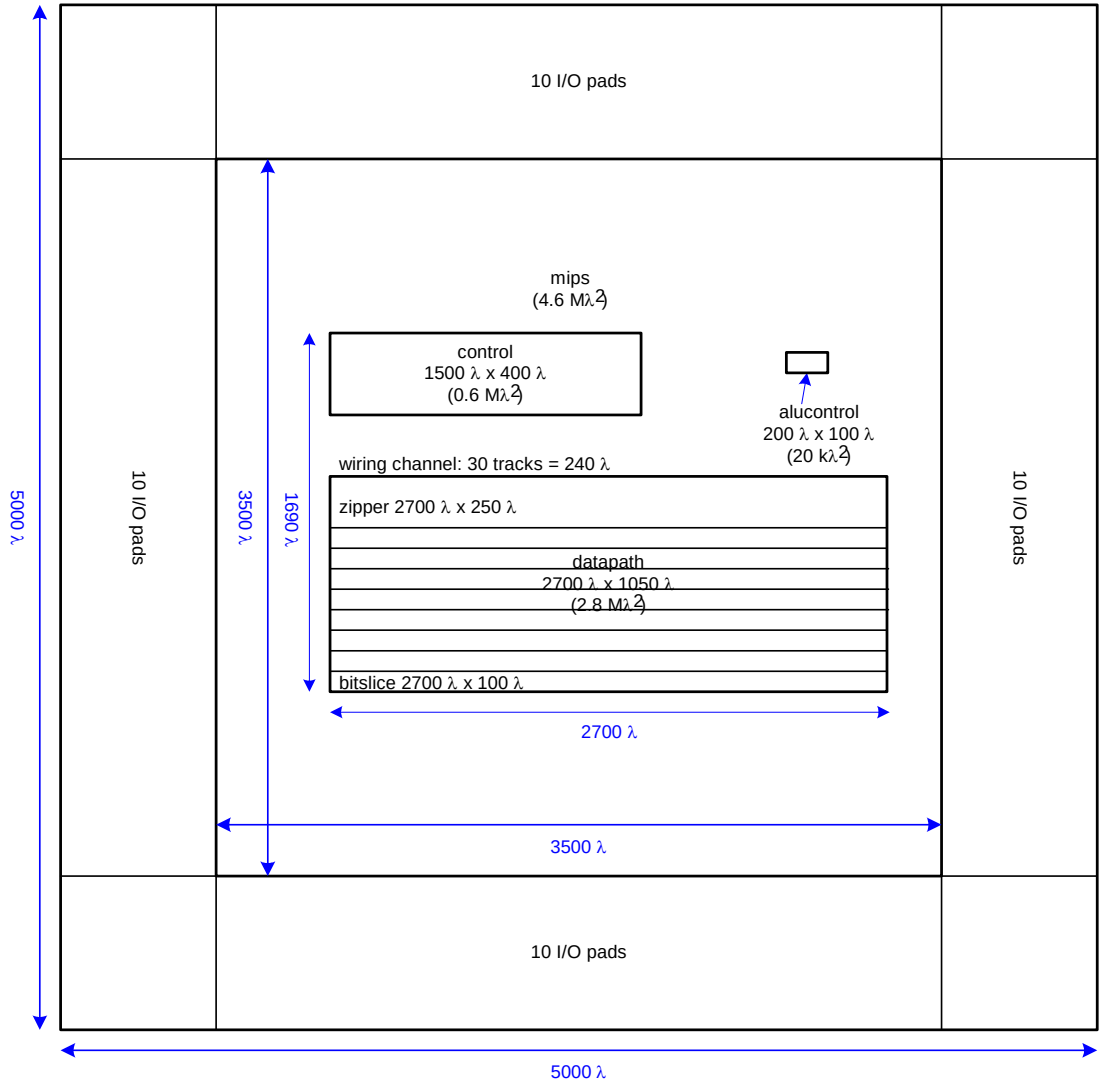
# MIPS: Block Diagram

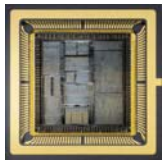




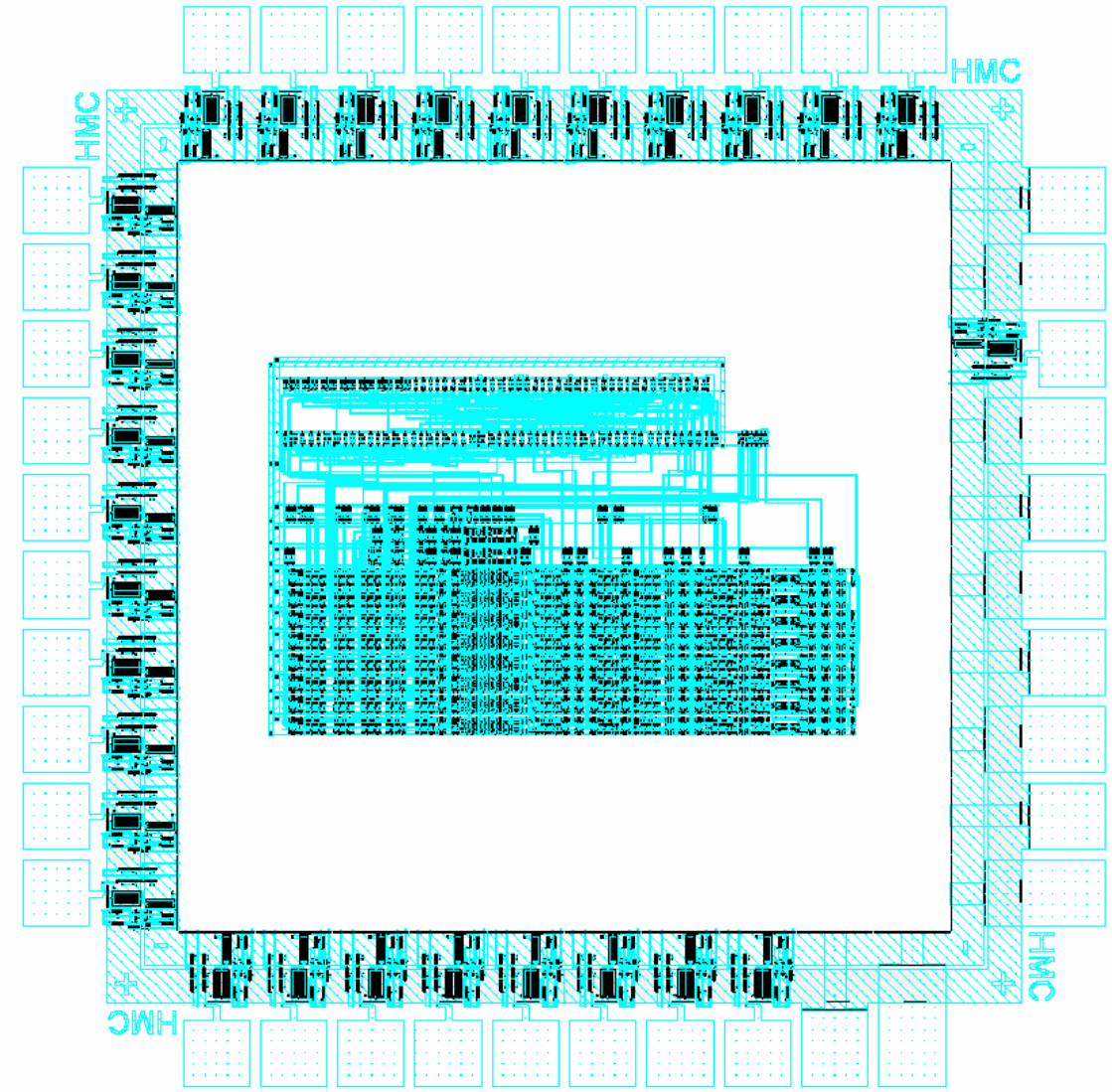


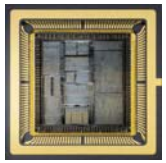
# MIPS: Floorplan





# MIPS : Layout



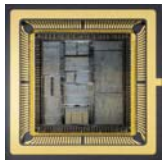


# Area Estimation

- Need area estimates to make floorplan
  - Compare to another block you already designed
  - Or estimate from transistor counts
  - Budget room for large wiring tracks
  - Your mileage may vary!

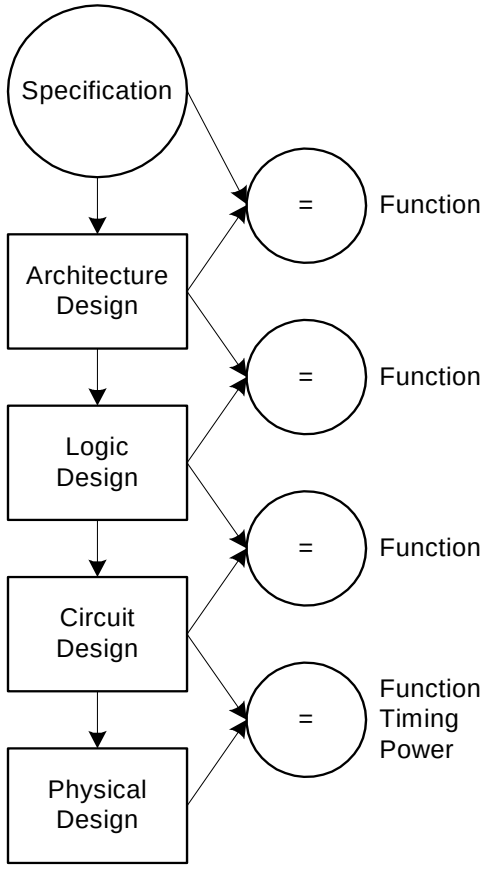
**Table 1.10 Typical layout densities**

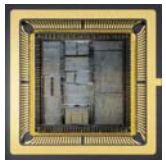
| Element | Area |
|---|---|
| random logic (2-level metal process) | 1000 – 1500 $\lambda^2$ / transistor |
| datapath | 250 – 750 $\lambda^2$ / transistor<br>or 6 WL + 360 $\lambda^2$ / transistor |
| SRAM | 1000 $\lambda^2$ / bit |
| DRAM (in a DRAM process) | 100 $\lambda^2$ / bit |
| ROM | 100 $\lambda^2$ / bit |



# Design Verification

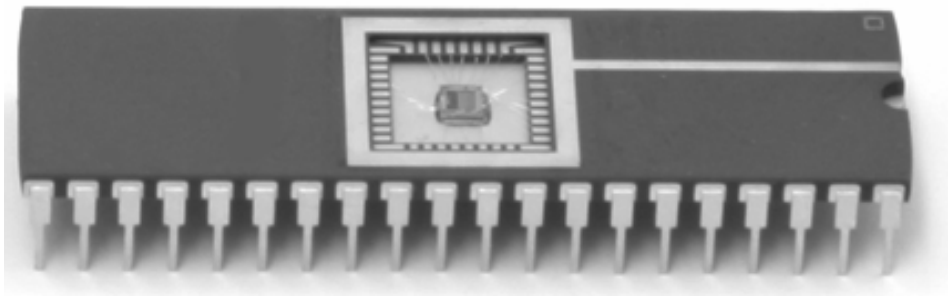
- Fabrication is slow & expensive
  - MOSIS 0.6μm: $1000, 3 months
  - State of art: $1M, 1 month
- Debugging chips is very hard
  - Limited visibility into operation
- Prove the design before building!
  - Logic simulation
  - Ckt. simulation / formal verification
  - Layout vs. schematic comparison
  - Design & electrical rule checks
- Verification is > 50% of effort on most chips!

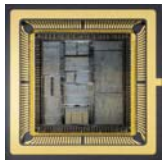




# Fabrication & Packaging

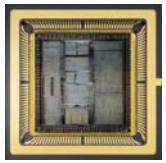
- Tapeout final layout
- Fabrication
  - 6, 8, 12” wafers
  - Optimized for throughput, not latency (10 weeks!)
  - Cut into individual dice
- Packaging
  - Bond gold wires from die I/O pads to package





# Testing

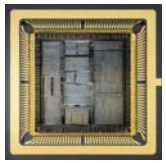
- Test that chip operates
  - Design errors
  - Manufacturing errors
- A single dust particle or wafer defect kills a die
  - Yields from 90% to < 10%
  - Depends on die size, maturity of process
  - Test each part before shipping to customer



# Microwind and DSCH

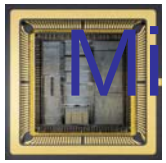
- Microwind is a tool for designing and simulating circuits at layout level. The tool features full editing facilities (copy, cut, past, duplicate, move), various views (MOS characteristics, 2D cross section, 3D process viewer), and an analog simulator.
- DSCH is a software for logic design. Based on primitives, a hierarchical circuit can be built and simulated. It also includes delay and power consumption evaluation.

Source: Microwind Based Design (http://vsp2.ecs.umass.edu/vspg/658/TA_Tools/microwind/Microwind.html)



# Microwind and DSCH …

- Download the followings from http://www.microwind.org/
  - Microwind3
  - DSCH3
  - User Manual: http://intrage.insa-toulouse.fr/~etienne/microwind/manual_lite.pdf
- Installation and Use:
  - Unzip the files above to be able to work with Microwind.
  - Read the reference manual for the software.
  - Double click on Microwind3.exe to start the layout editor or on Dsch3.exe to start the schematic editor.

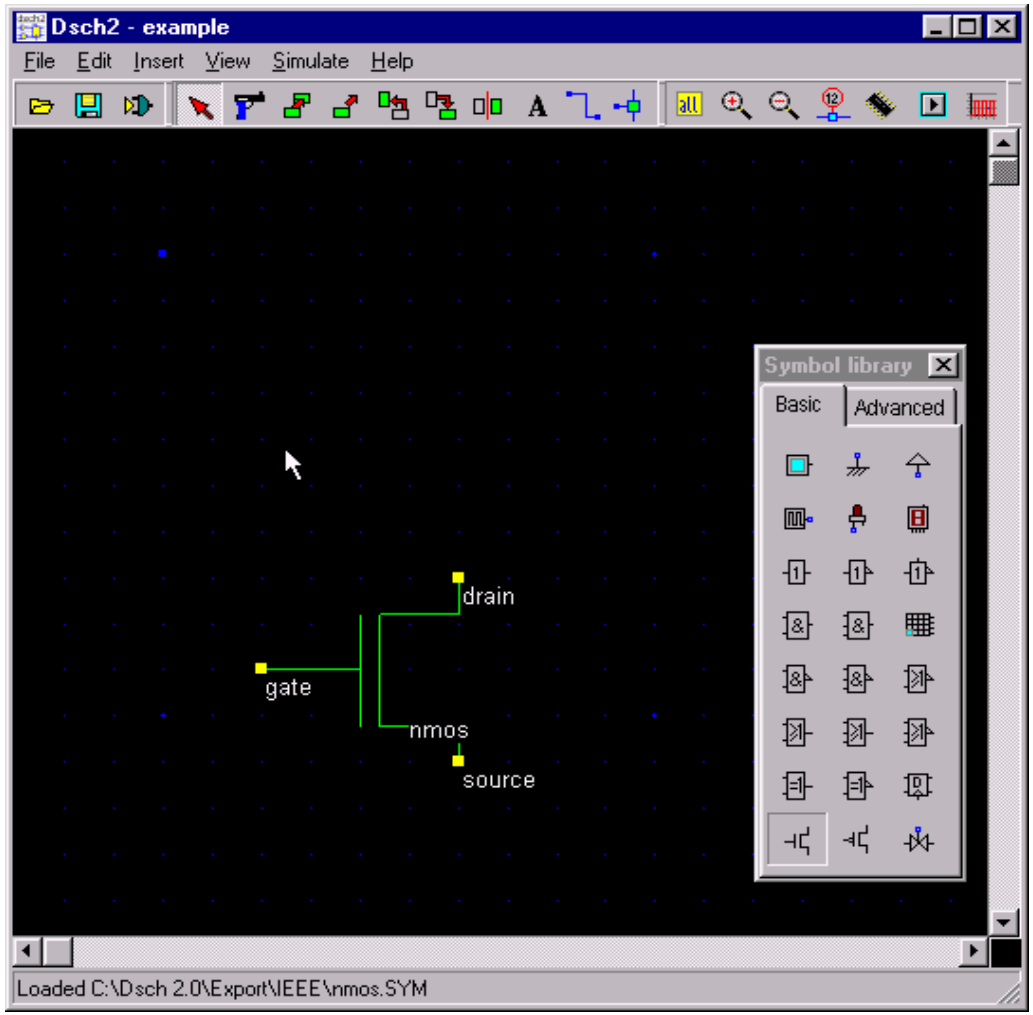


# Microwind and DSCH : NOR Example

- We will learn both the design flow and the CAD tools.
- The specifications we are going to see may be different for different foundry and technology.
- Design Example (3 Levels) : NOR Gate
  - Logic Design
  - Circuit Design
  - Layout Design

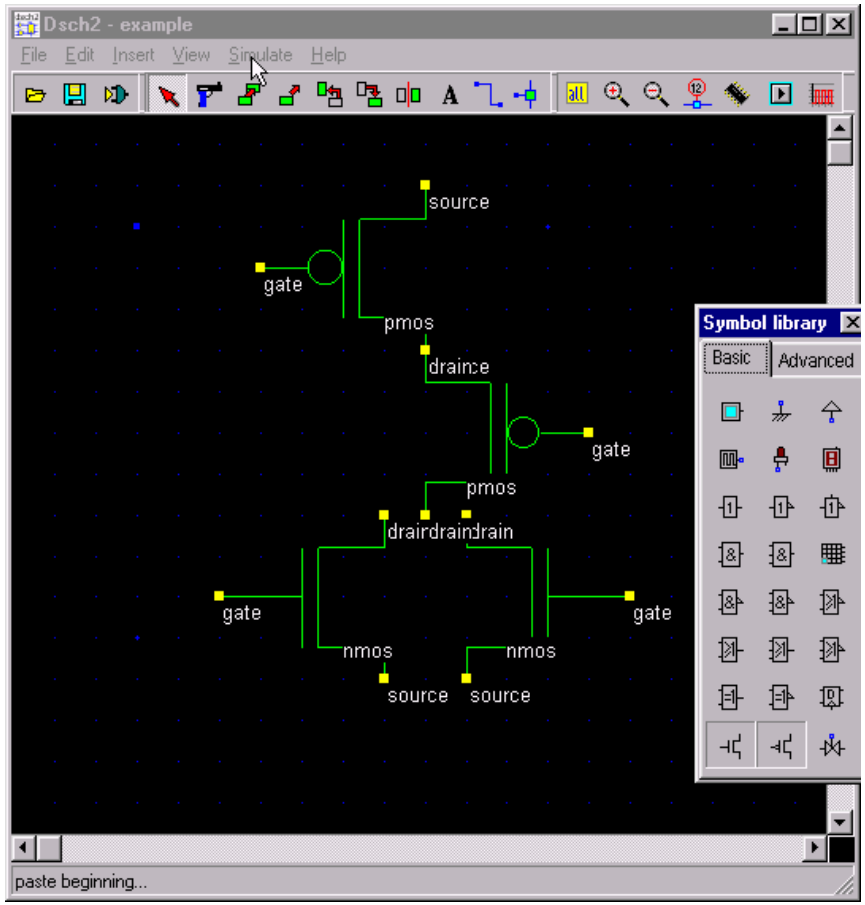
# Microwind / DSCH NOR Example: Logic

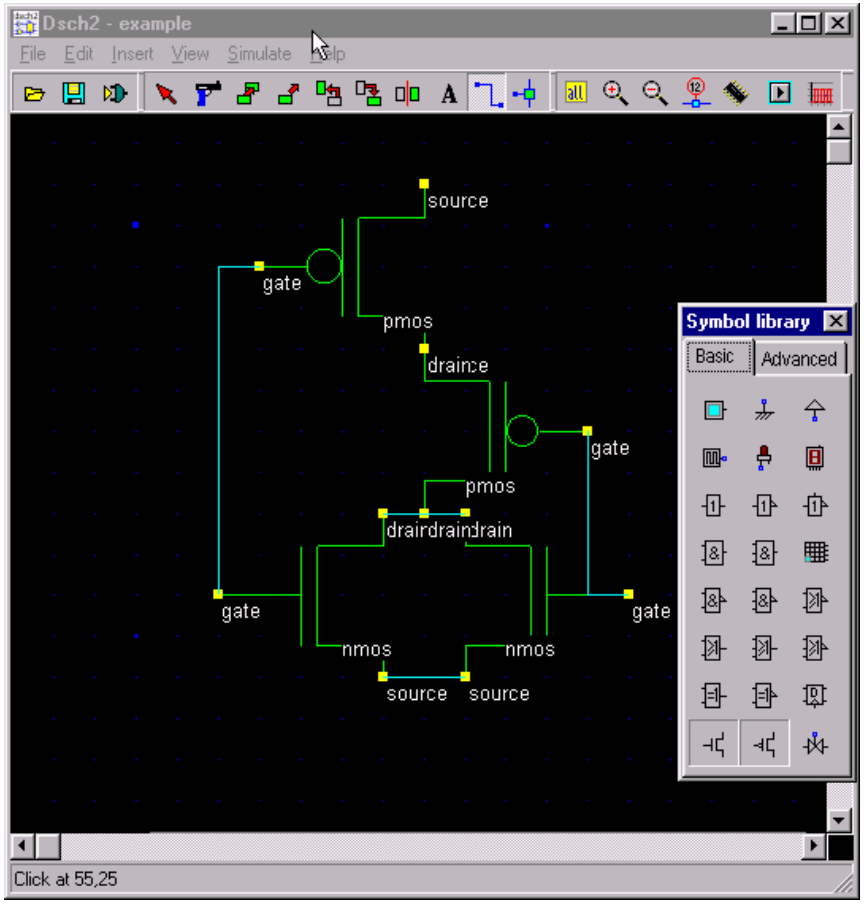
- Open the Schematic Editor in Microwind (DSCH3). Click on the transistor symbol in the Symbol Library on the right.
- Instantiate NMOS or PMOS transistors from the symbol library and place them in the editor window.



# Microwind / DSCH NOR Example: Logic

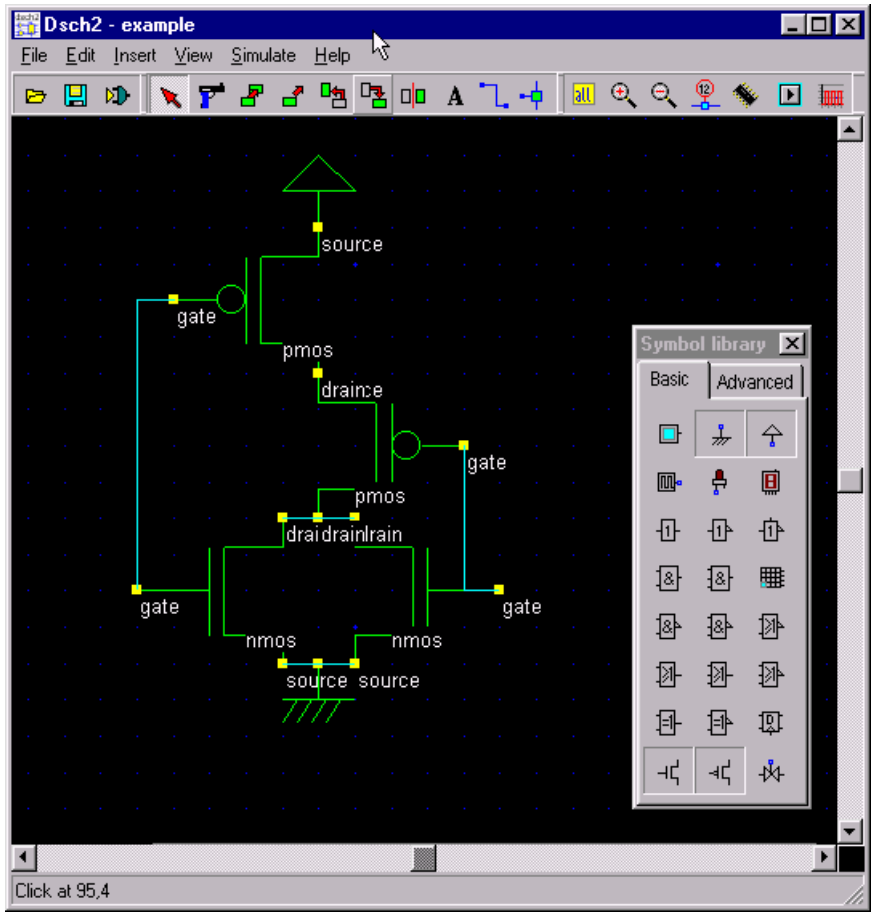
- Instantiate 2 NMOS and 2 PMOS transistors.
- Connect the drains and sources of transistors.

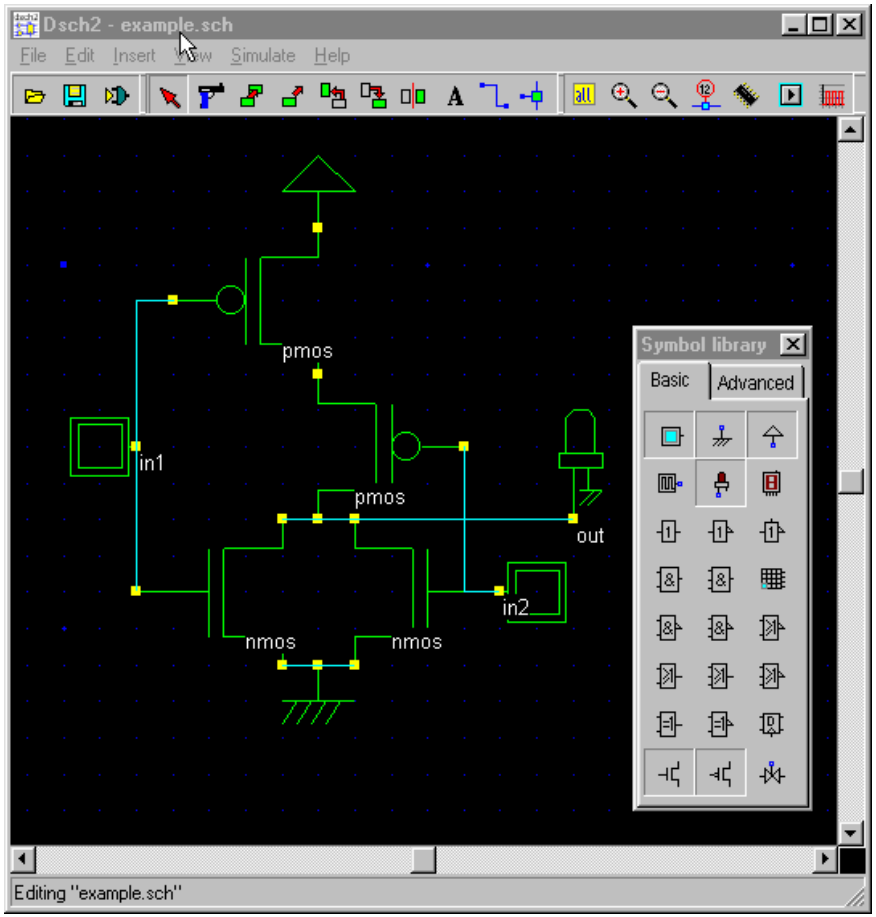


# Microwind / DSCH NOR Example: Logic

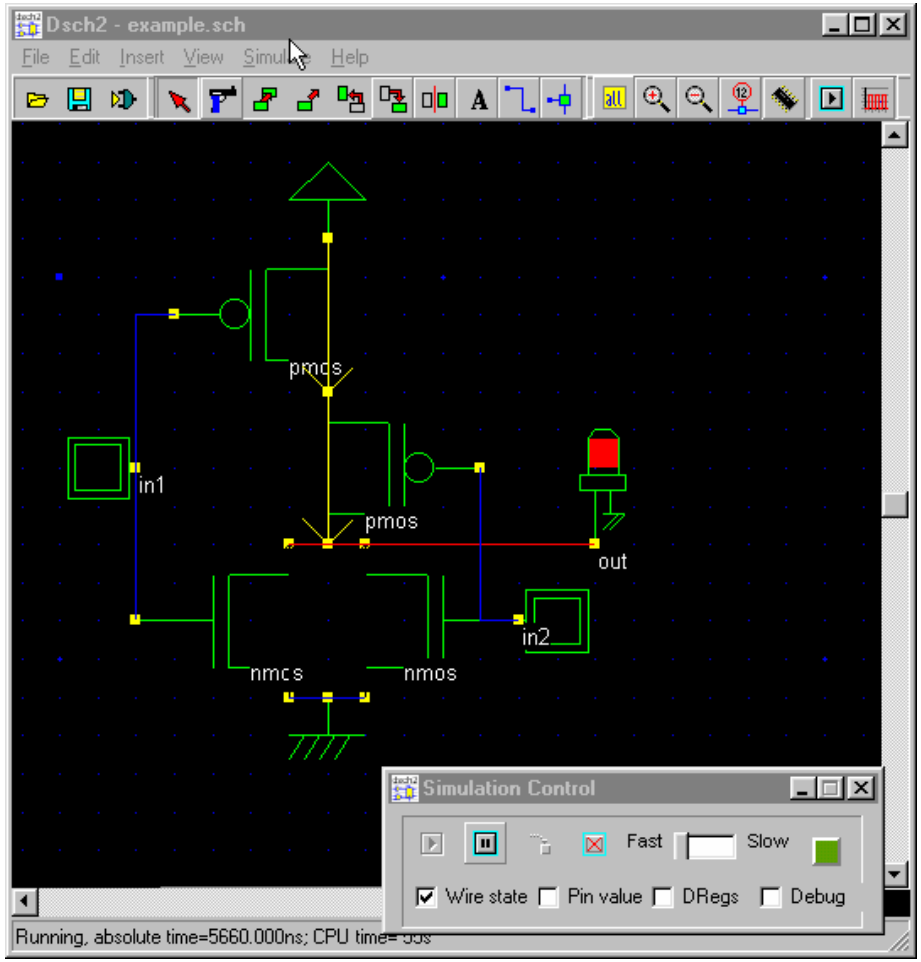
- Connect Vdd and GND to the schematic.
- Connect input button and output LED.



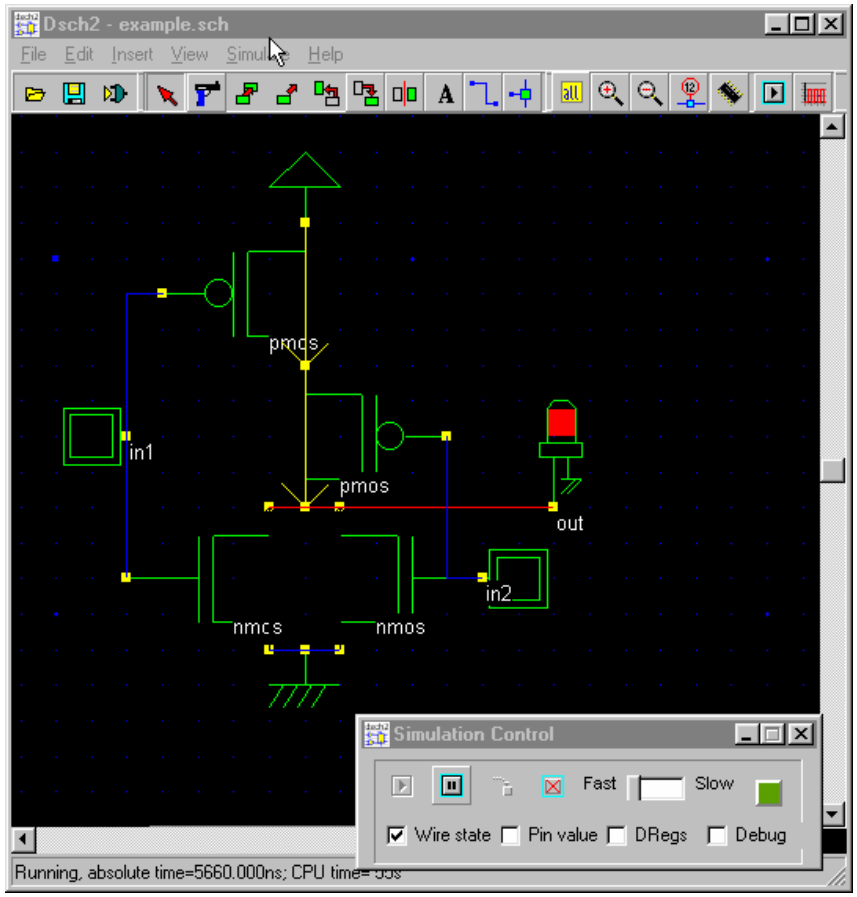
# Microwind / DSCH NOR Example: Logic

- You now have NOR schematic ready.
- Use your logic simulator to verify the functionality of your schematic.
- The next step is to simulate the circuit and check for functionality.
- Click on, Simulate -> Start simulation.
- This brings up a Simulation Control Window.
- Click on the input buttons to set them to 1 or 0. Red color in a switch indicates a '1'.

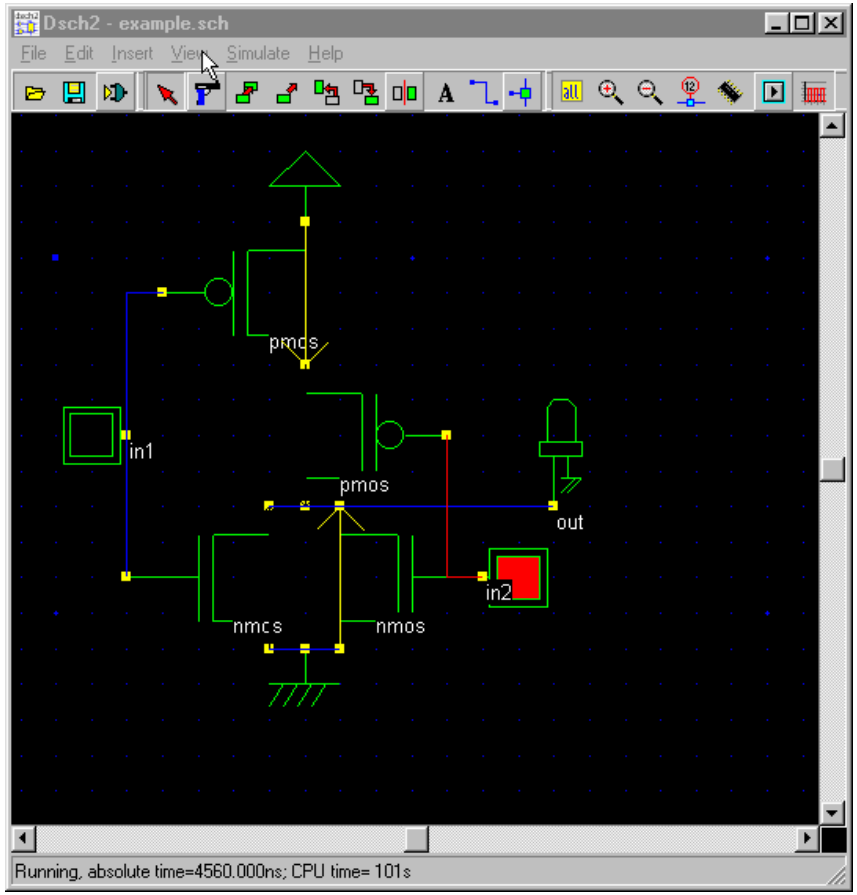


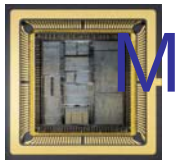
# Microwind / DSCH NOR Example: Logic

- Inputs: 0 0



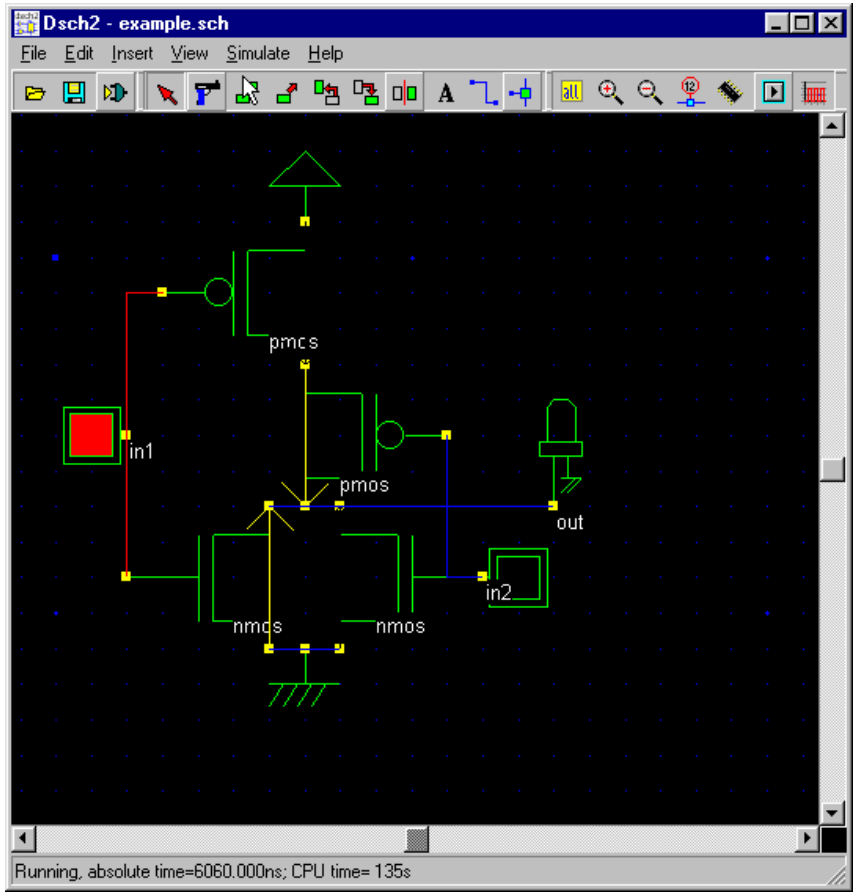

- Inputs: 0 1



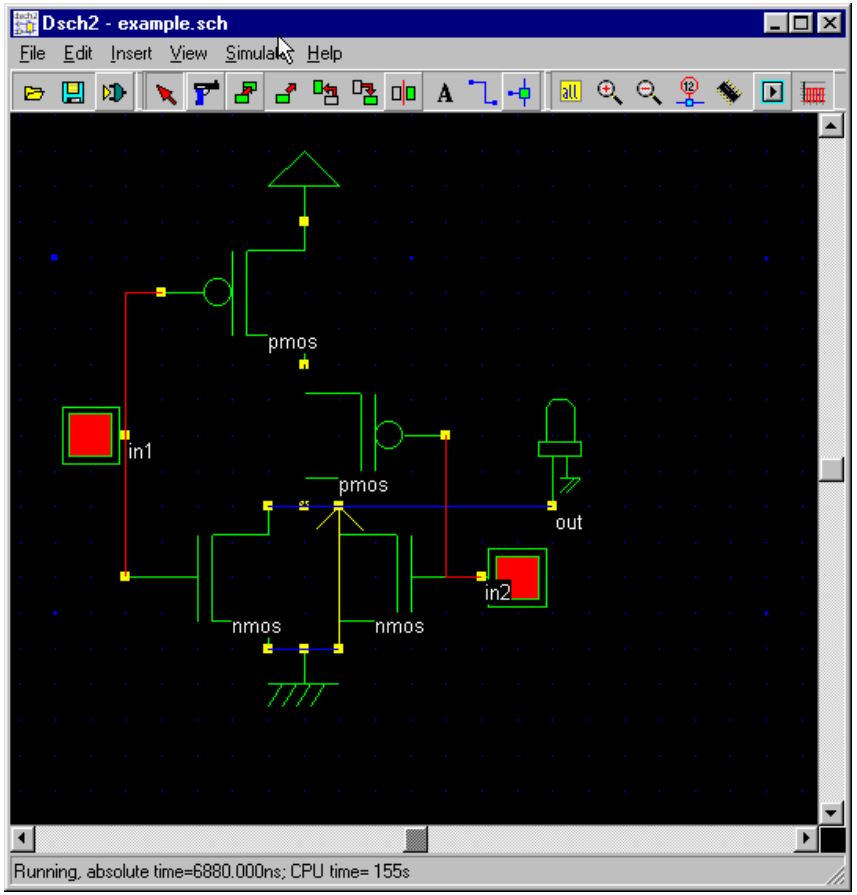


# Microwind / DSCH NOR Example: Logic

- Inputs: 1 0

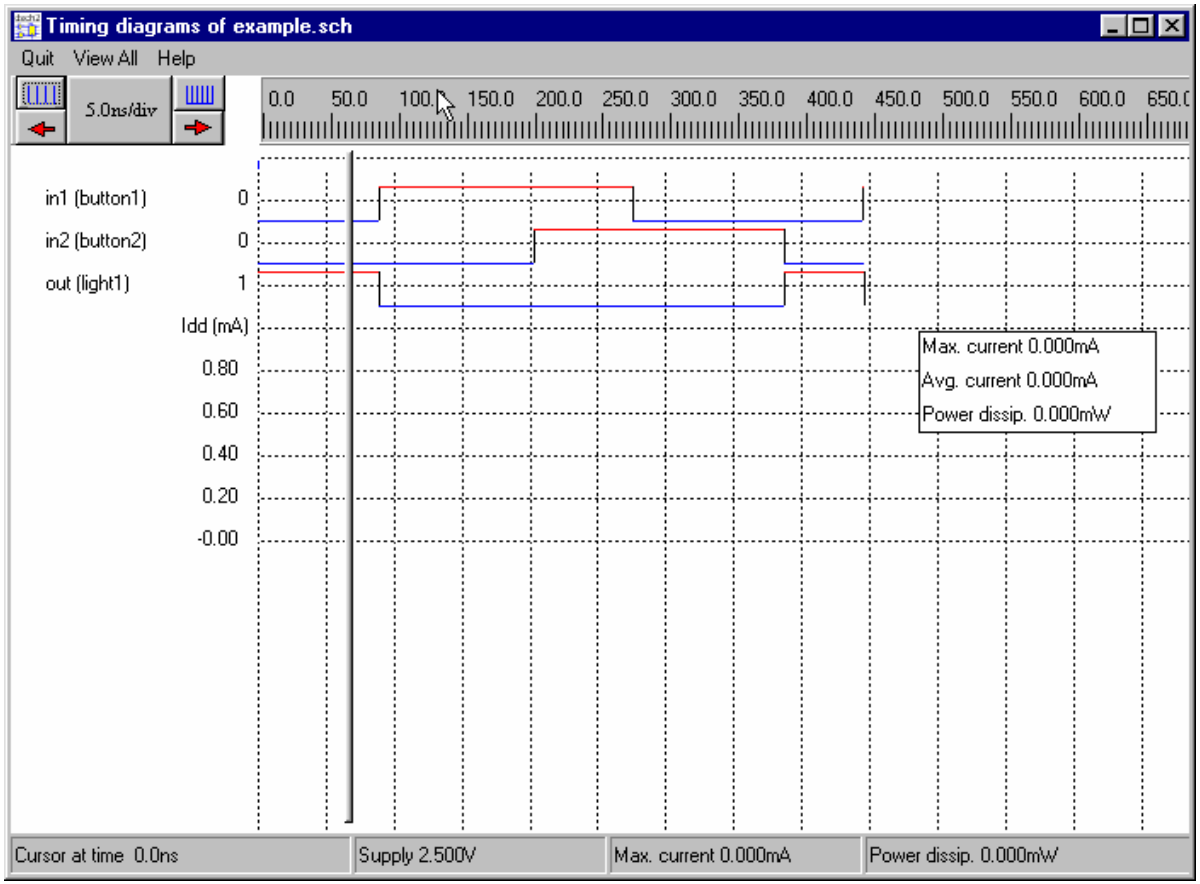



- Inputs: 1 1



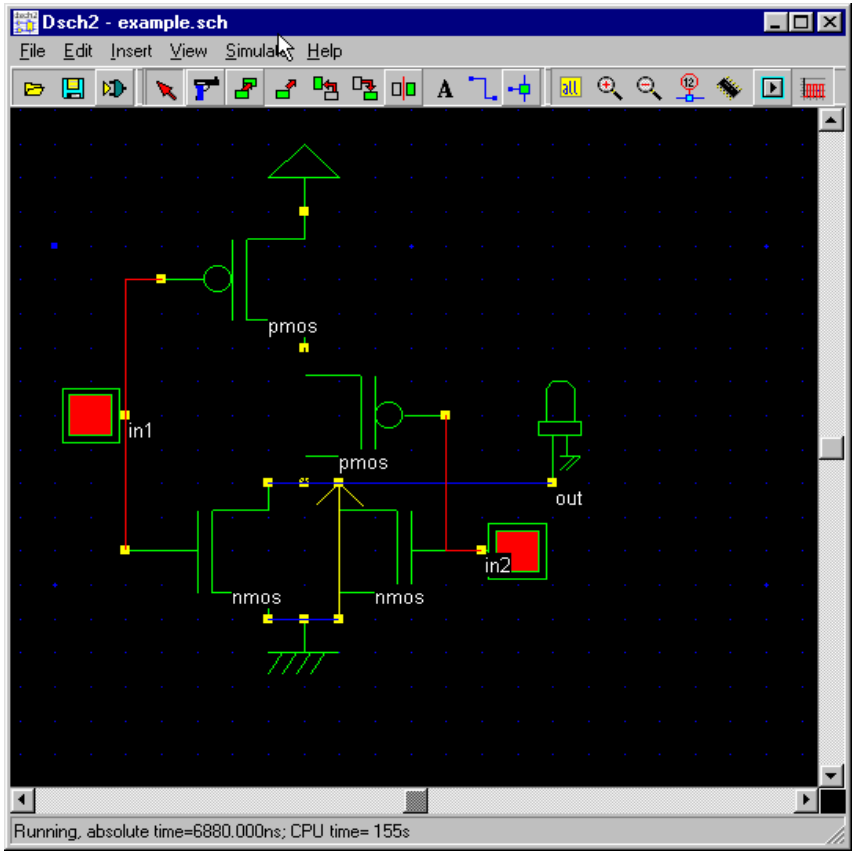
# Microwind / DSCH NOR Example: Logic

- The simulation output can be observed as a waveform after the application of the inputs as above. Click on the timing diagram icon in the icon menu to see the timing diagram of the input and output waveforms.

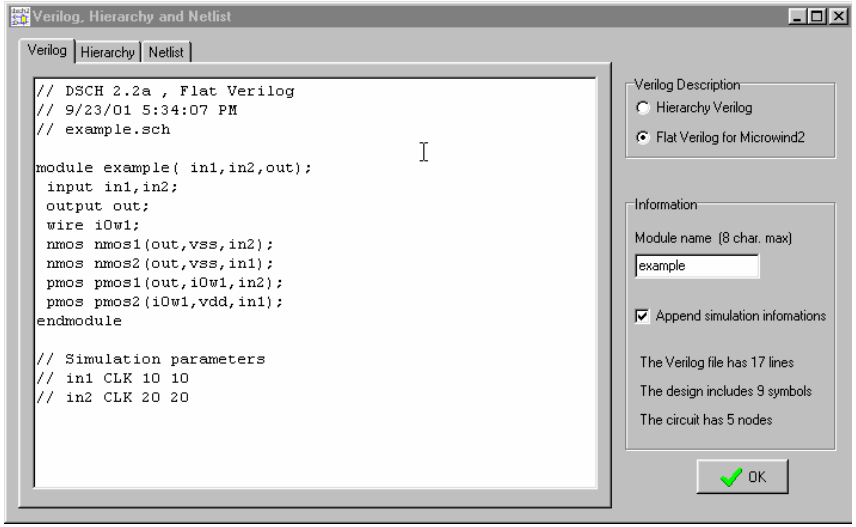


# Microwind / DSCH NOR Example: Circuit

- Simulate your system with your hand calculated transistor sizes.

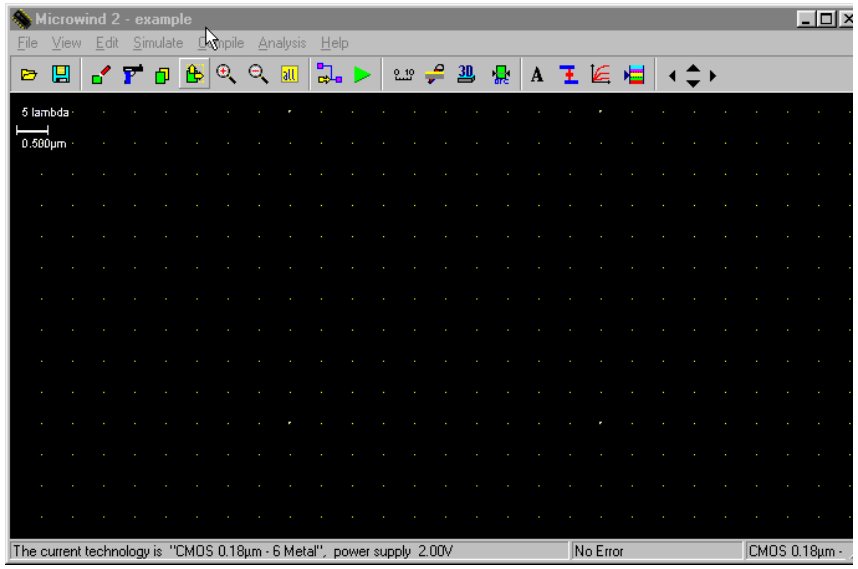


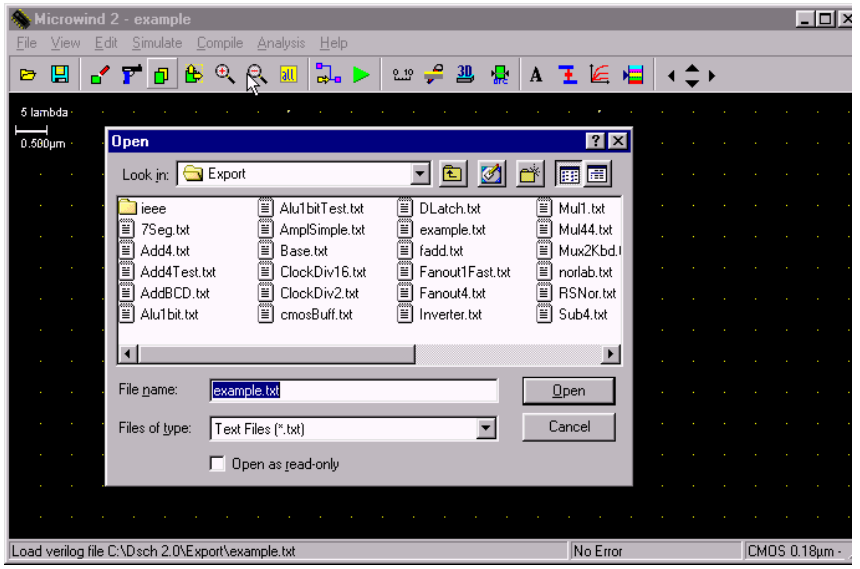

- Click *File -> Make Verilog File*. The *Verilog, Hierarchy and Netlist* window appears. This window shows the verilog representation of NOR gate. Click OK to save the Verilog as a .txt file.



# Microwind / DSCH NOR Example: Circuit

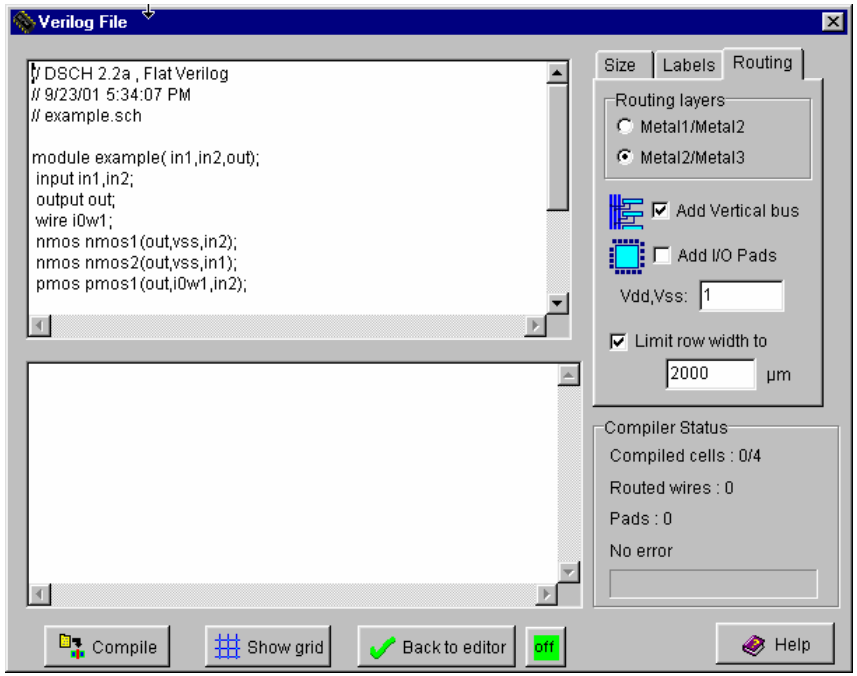
- Open the layout editor window in Microwind. Click *File* -> *Select Foundry* and select *X.rul*. This sets your layout designs in X technology.
- Click on *Compile* -> *Compile Verilog File.* An *Open* Window appears. Select the .txt verilog file saved before and open it.

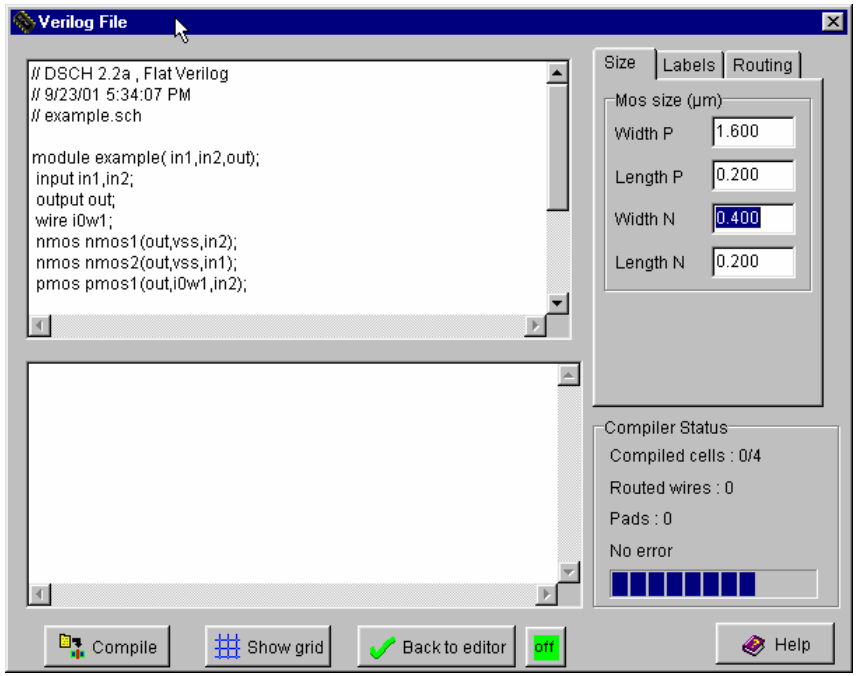


# Microwind / DSCH NOR Example: Circuit

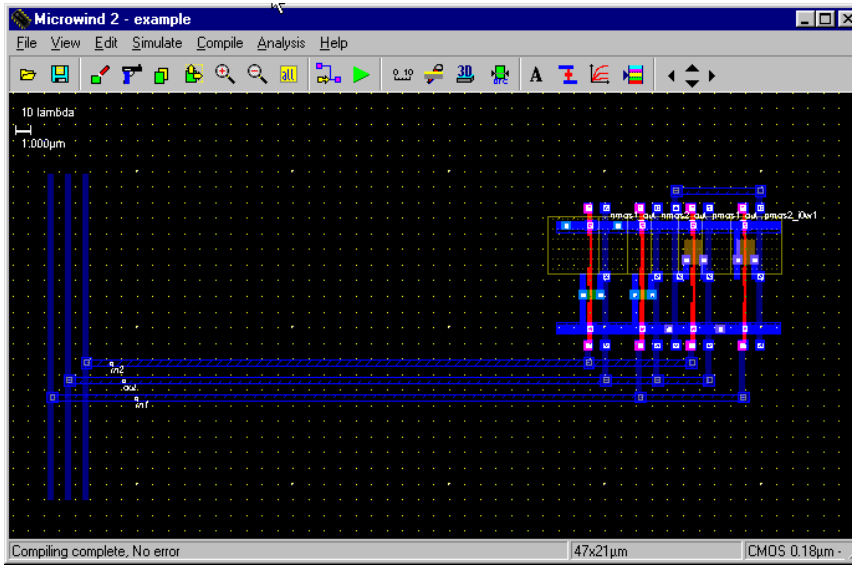
- After selecting the .txt file, a new window appears called Verilog file.
- Click on Size on the right top menus. This shows up the NMOS and PMOS sizes. Set the sizes according to choice.

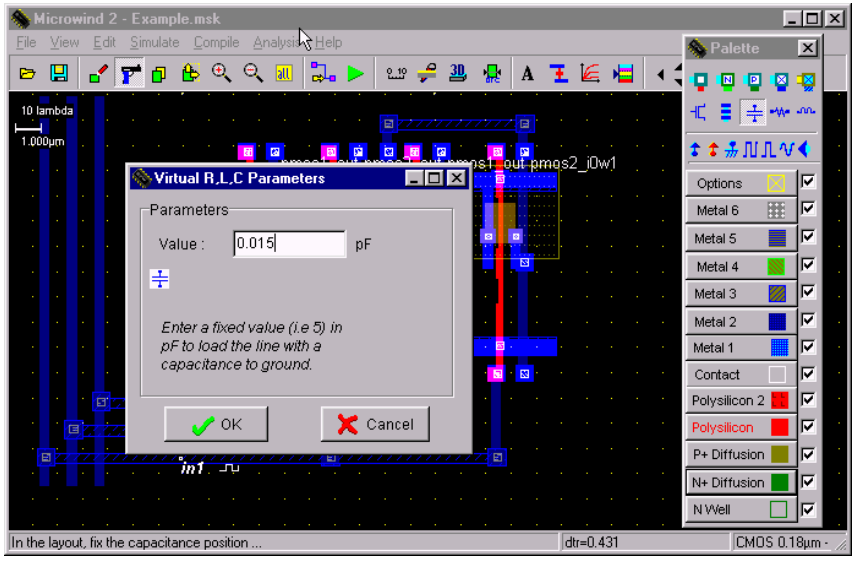


# Microwind / DSCH NOR Example: Circuit

- Click Compile and then Back to editor in the Verilog File Window. This creates a layout in layout editor window using automatic layout generation procedure.
- Add a capacitance to the output of the design. The value of the capacitance depends on your choice.

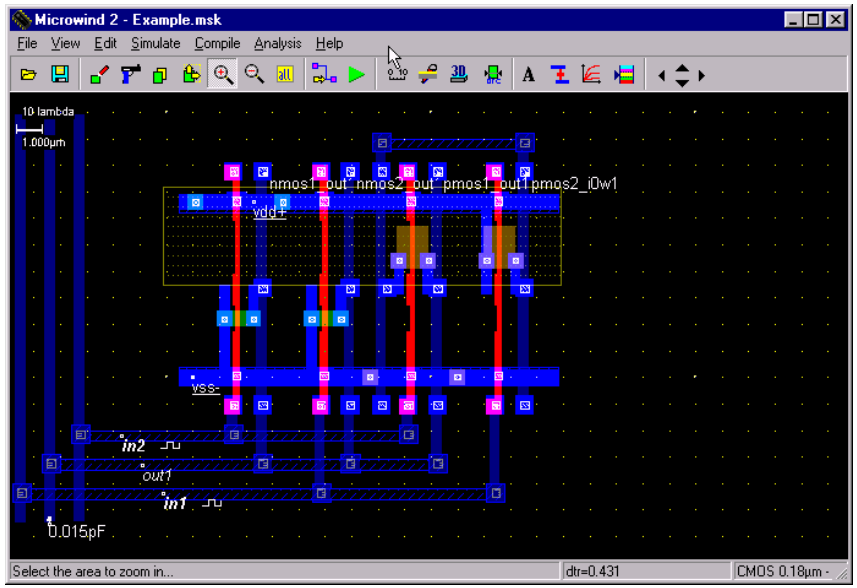


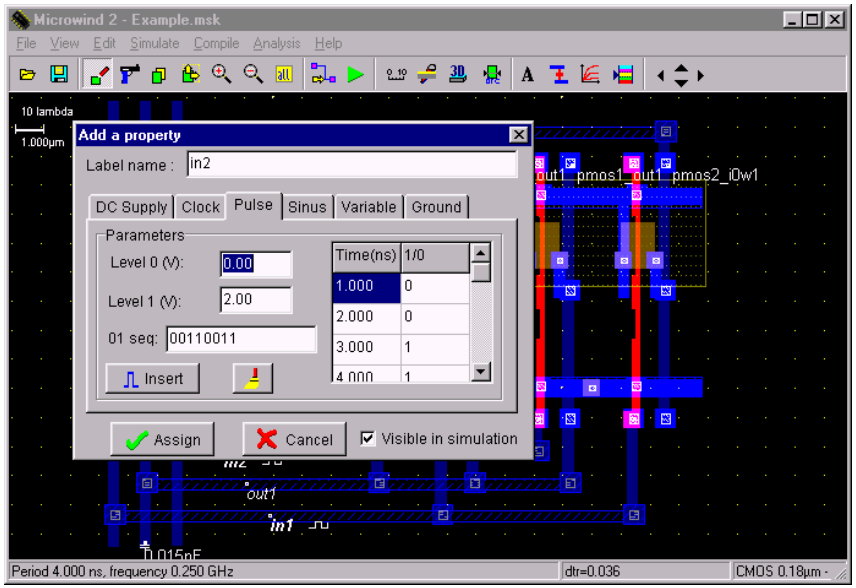



# Microwind / DSCH NOR Example: Circuit

- Click on OK. The capacitance is shown on the left bottom corner with a value of 0.015fF.

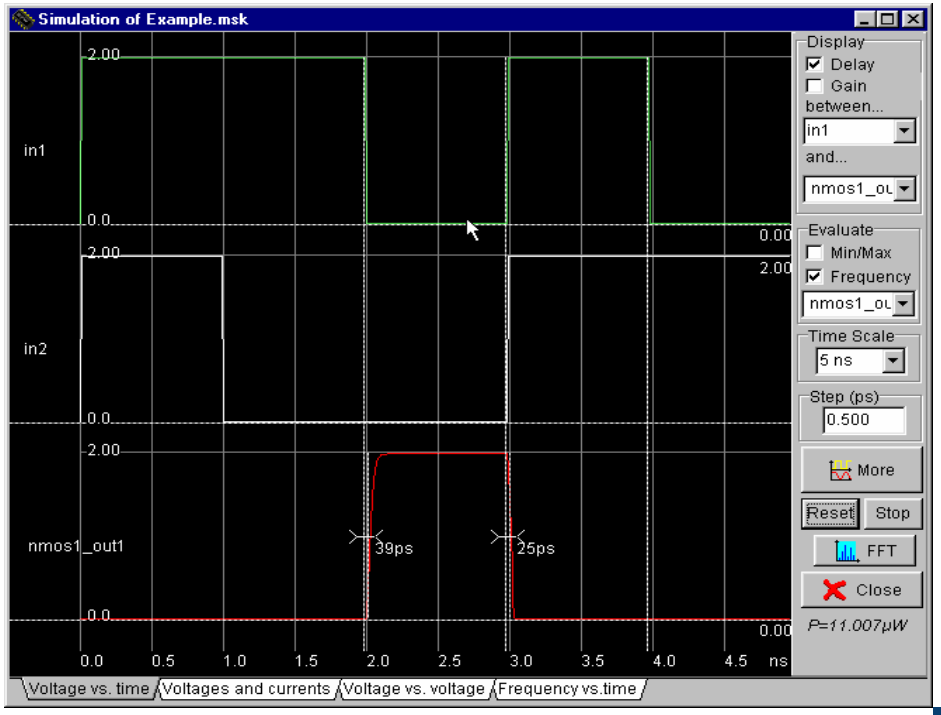
- Click on the label marked In1. A window appears. Click on the *Pulse* option in the window. Insert a 01 sequence for that specific input and click on *Insert*. Then click on *Assign*. Perform this assignment on the other inputs.



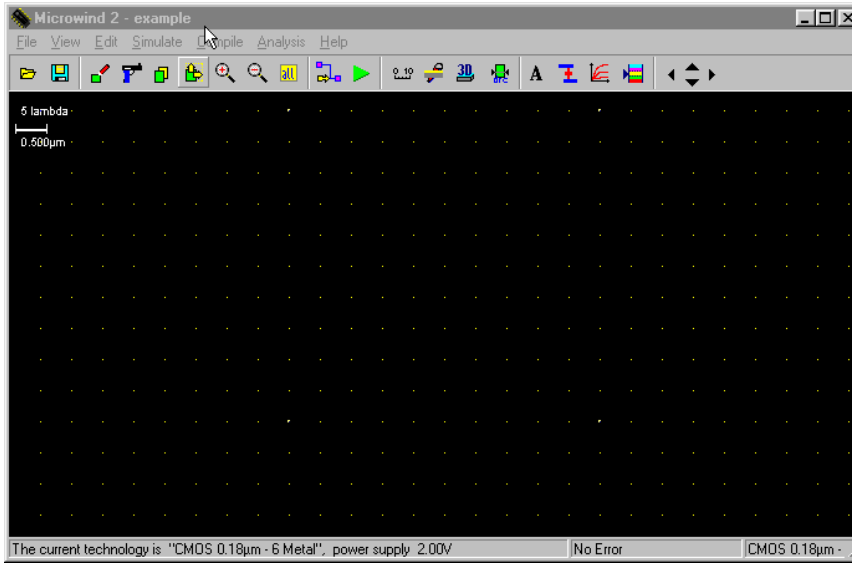
# Microwind / DSCH NOR Example: Circuit

- Click *Simulate -> Run simulation*. A simulation window appears with inputs and output, shows the tphl, tplh and tp of the circuit. The power consumption is also shown on the right bottom portion of the window.
- If you are unable to meet the specifications of the circuit change the transistor sizes. Generate the layout again and run the simulations till you achieve your target delays.

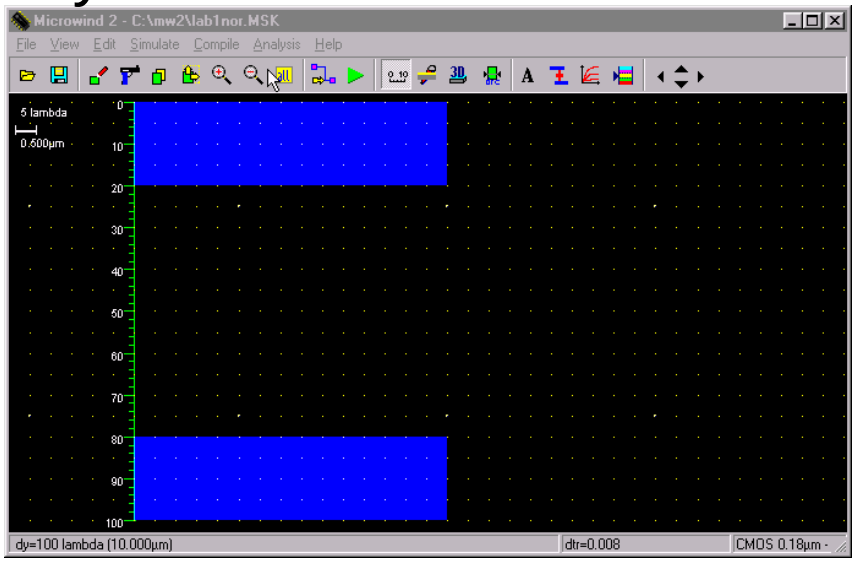


# Microwind / DSCH NOR Example: Layout

- Design the layout manually
- Open the layout editor window in Microwind. Click *File -> Select Foundry* and select *X.rul*

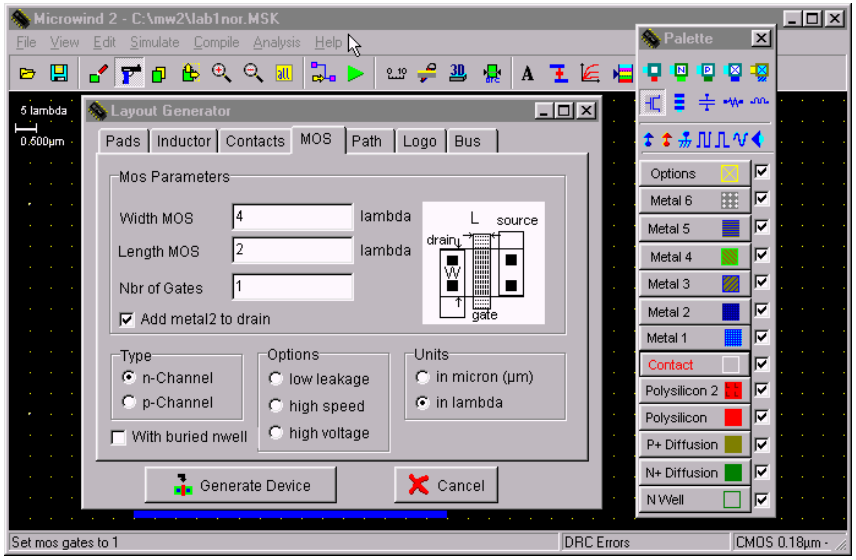


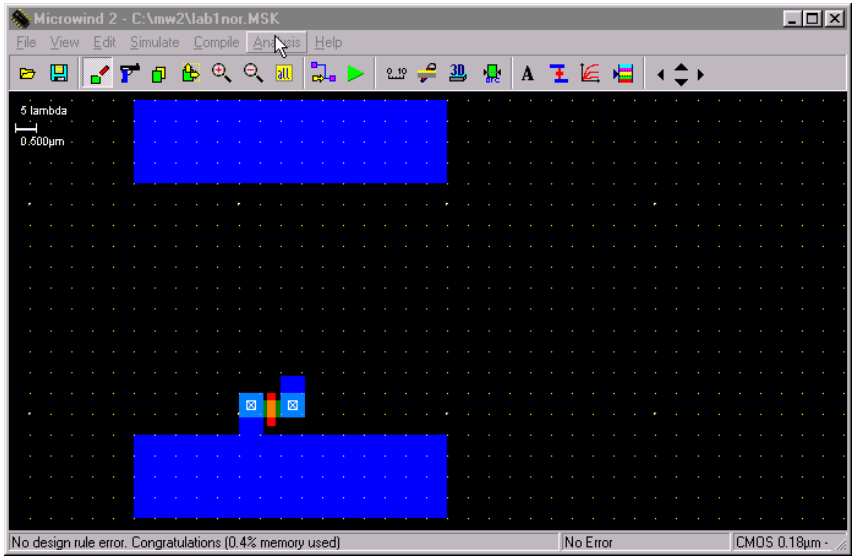

- Vdd and GND rails are of Metal1. The top rail is used as Vdd and the bottom one as GND. Click on Metal 1 in the palette and then create the required rectangle in the layout window.



# Microwind / DSCH NOR Example: Layout

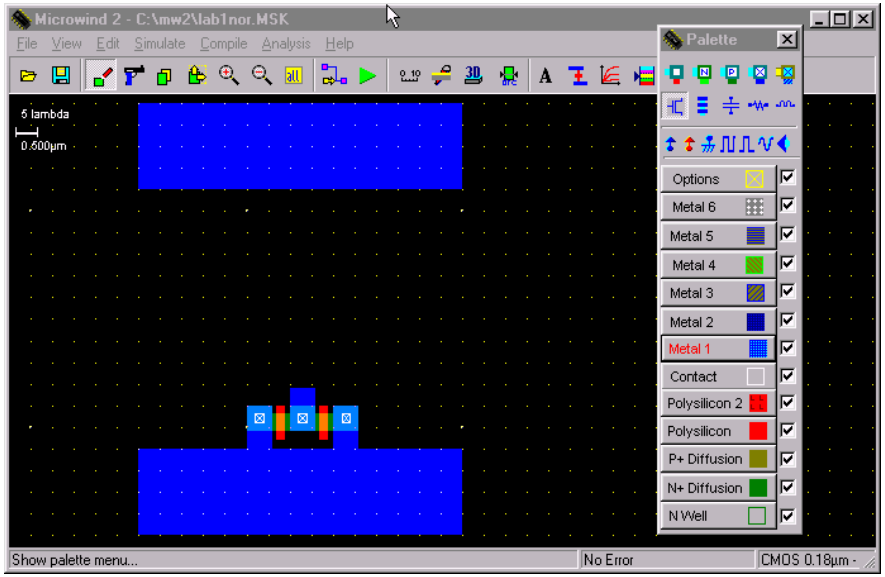
- The next step is to build the NMOS transistors. Click on the transistor symbol in the palette. Set the W, L of the transistor.
- Then click on Generate device. The source of the transistor is connected to the GND rail.

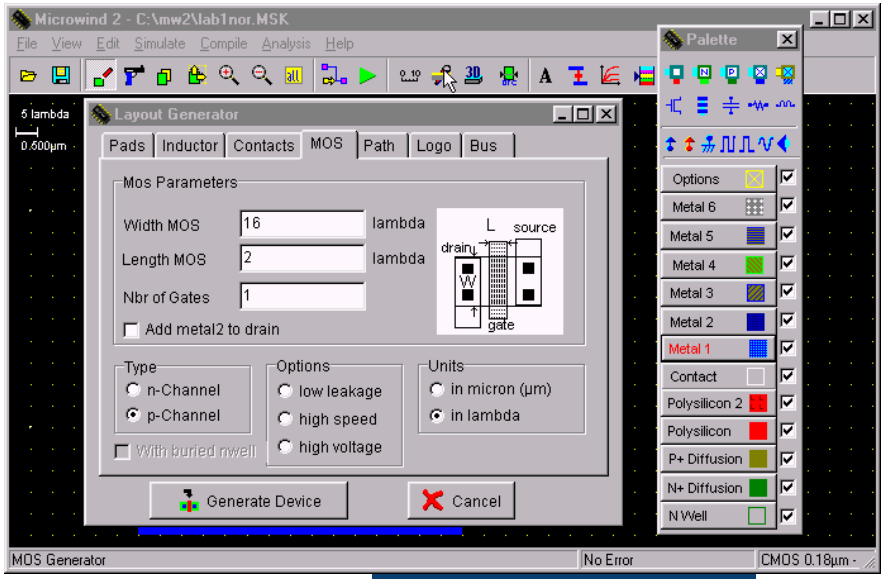


# Microwind / DSCH NOR Example: Layout

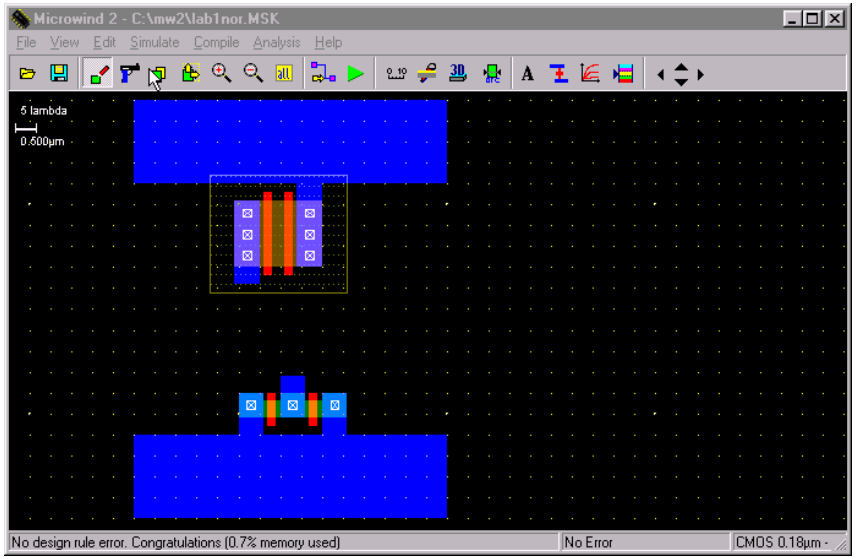
- Create another NMOS and place it in parallel to the first NMOS device. We share the two devices' drain diffusions. A DRC check can be run by clicking on *Analysis -> Design Rule Checker*.
- The next step is to place two PMOS transistors in series.

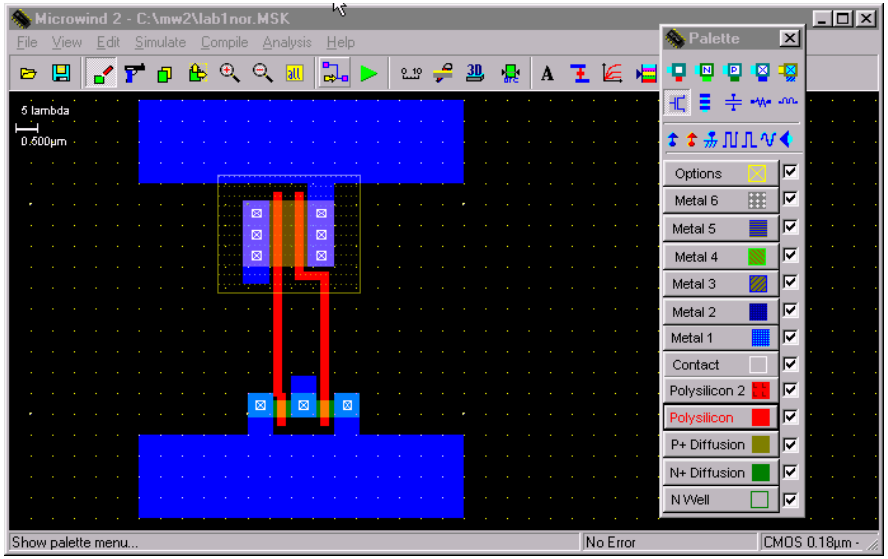


# Microwind / DSCH NOR Example: Layout

- Place the PMOs transistor on layout close to the Vdd rail on the top. To construct two PMOS transistors in series, diffusions are shifted to a side and another poly line is added as second transistor. The diffusion is shared to save area and reduce capacitance.
- The next step is to connect the inputs and the output of the two transistors.

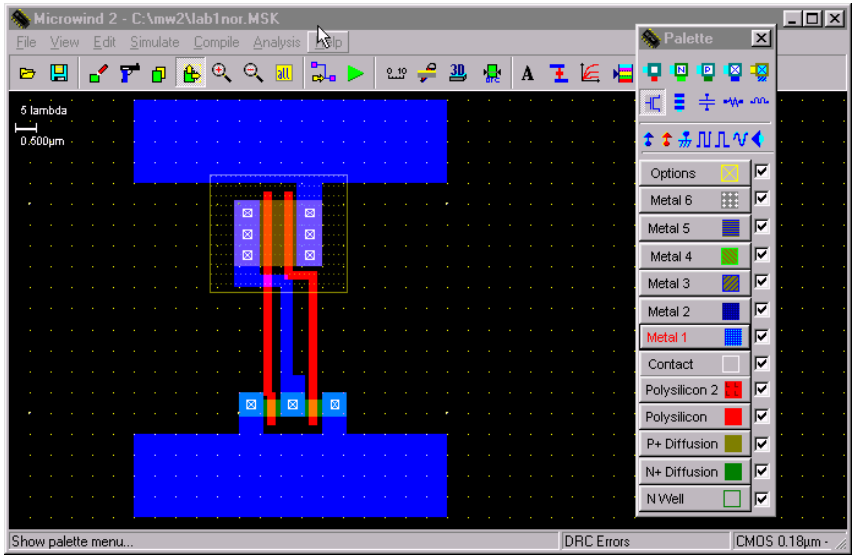


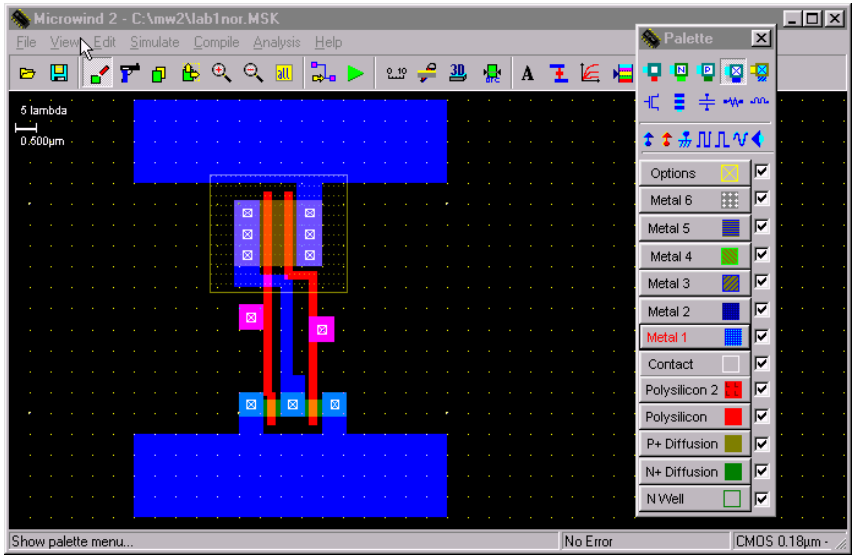



# Microwind / DSCH NOR Example: Layout

- Poly inputs are connected
- Metal output is connected.

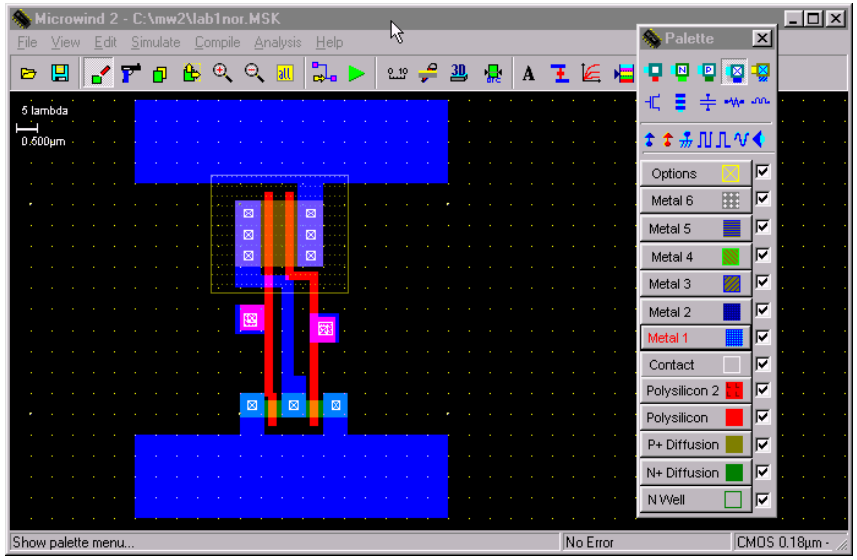
- The next step is to connect the poly to metal1 and then to metal2. The first symbol in the first row of the palette is the poly to metal1 contact.

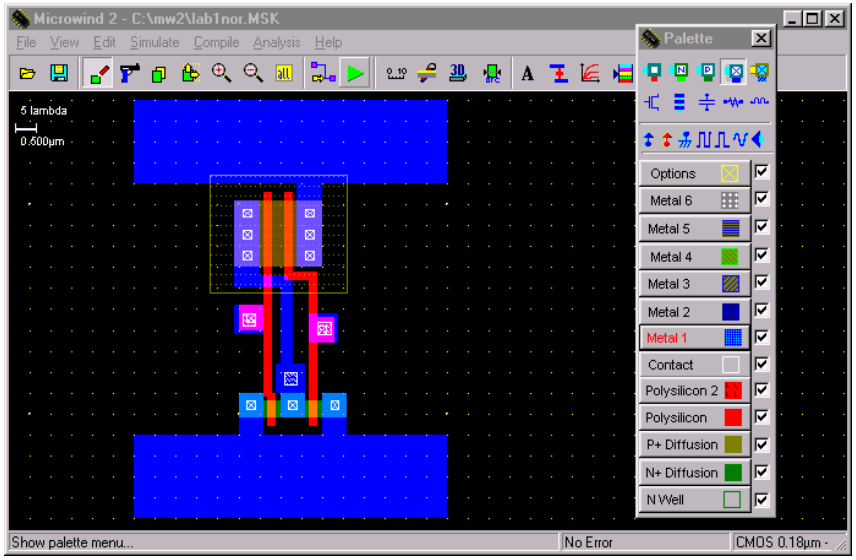


# Microwind / DSCH NOR Example: Layout

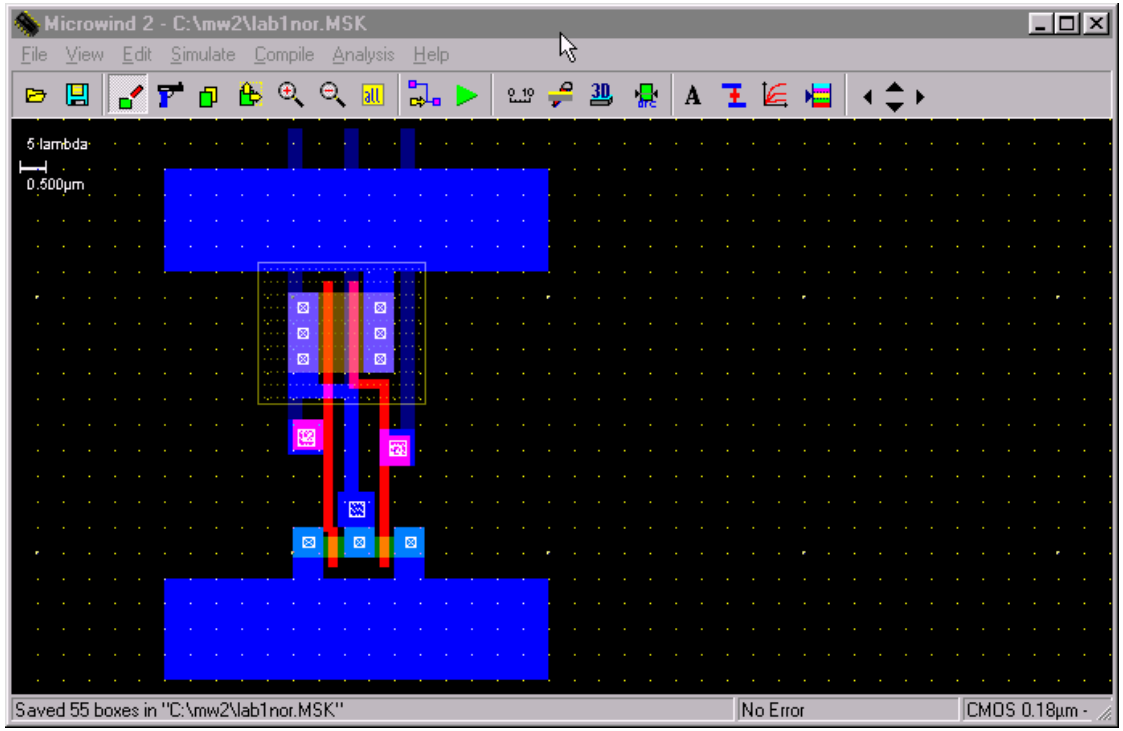
- Then we connect the metal1 to metal2 contact to the previous contact. This is the 4th contact on the first row.
- The next step is to connect the output Metal1 to Metal2. Once again use the 4th contact in the first row.



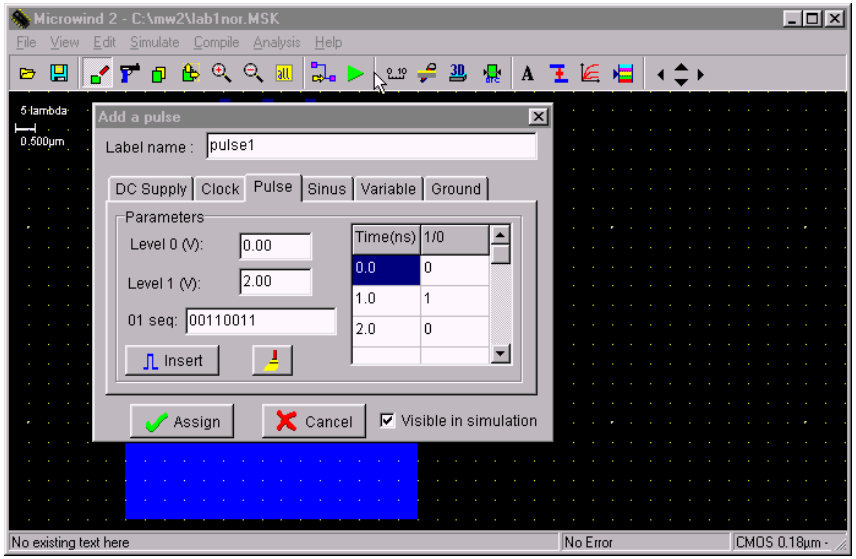
# Microwind / DSCH NOR Example: Layout

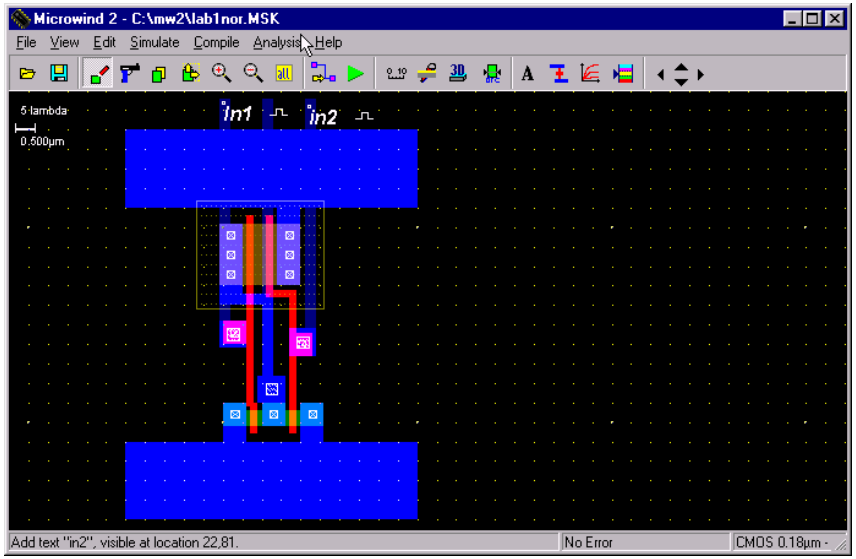
- Now we connect metal2 to the two inputs and one output and bring them to the top to go out of the cell.
- Observe the two inputs (left & right) and an output (middle) above the Vdd rail in dark blue color.



# Microwind / DSCH NOR Example: Layout

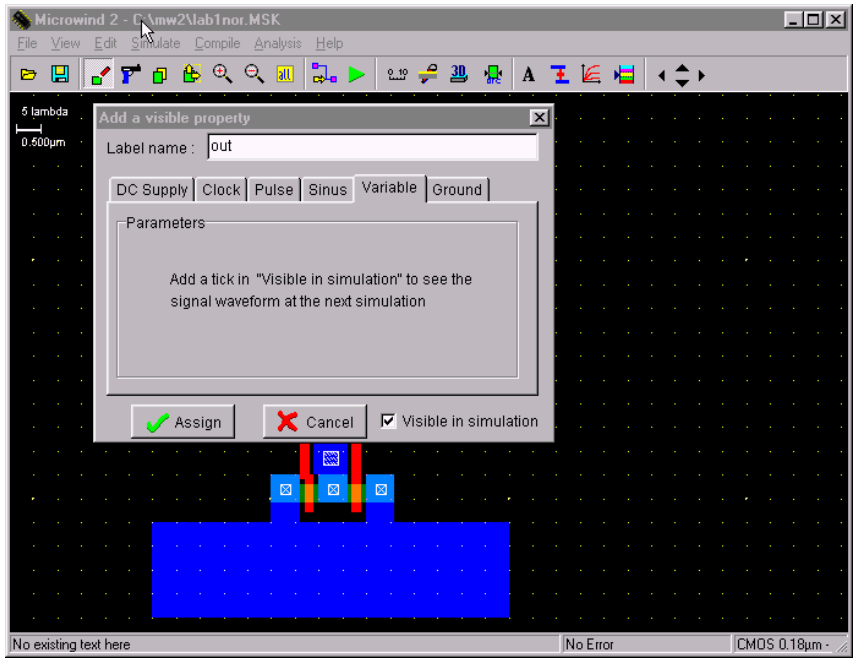
- Now we label the inputs and output as In1, In2 and out. Click on Add a Pulse Symbol in the palette (5th from the right in the 3rd row). Then click on the metal2 of one of the inputs. A window appears. Change the name of the input signal. Insert a 01 sequences and click on Insert. The click on Assign. Similarly assign the 2nd input a pulse.

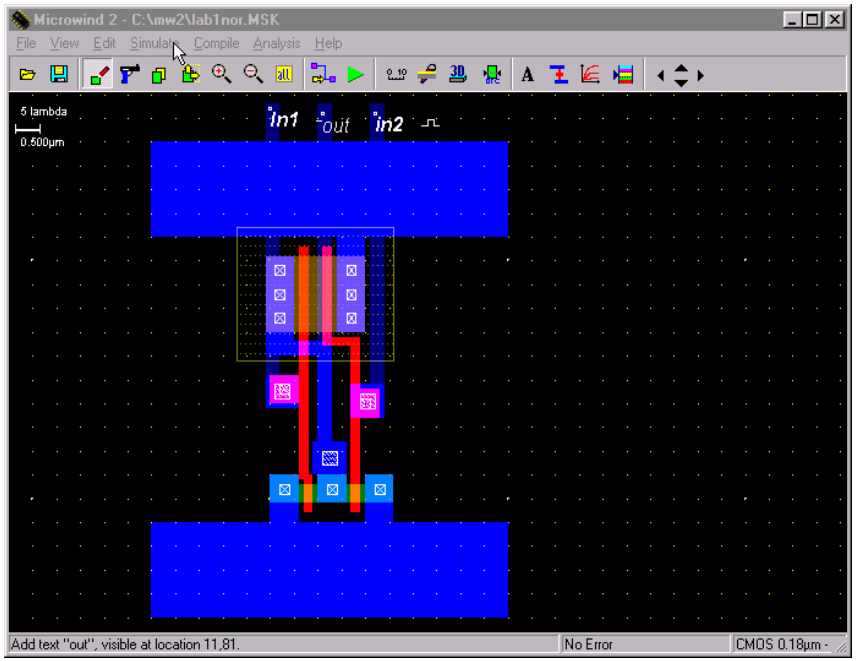


# Microwind / DSCH NOR Example: Layout

- Now select the Visible Node symbol from the palette (7th in the third row). Select it and click on the output. The 'Add a Visible Property' window appears. Change the label name to out. Select Visible in Simulation. Click on Assign. Now the output is also labeled.



# Microwind / DSCH NOR Example: Layout

- Select *Vdd Supply* and *GND* from the palette (third row). Also click on the capacitor (3rd in 2nd row) symbol and add it to the output. Also, extend the p-well into the Vdd Rail. The click on *Edit -> Generate -> Contacts*. Select PATH and then in Metal choose *Metal1 and N+ polarization*.

- To run the Simulation of your circuit, click on *Simulate -> Start Simulation*. Depending on the input sequences assigned at the input the output is observed in the simulation. The power value is also given.