

Lecture 6: Design Flow

CSCE 5730 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.



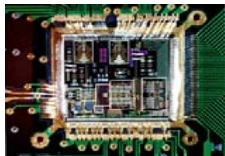
Lecture Outline

- Hierarchical Design
- Logic Design
- Circuit Design
- Physical Design



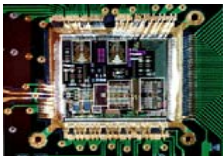
Hierarchical Design: Design Partitioning

- 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.

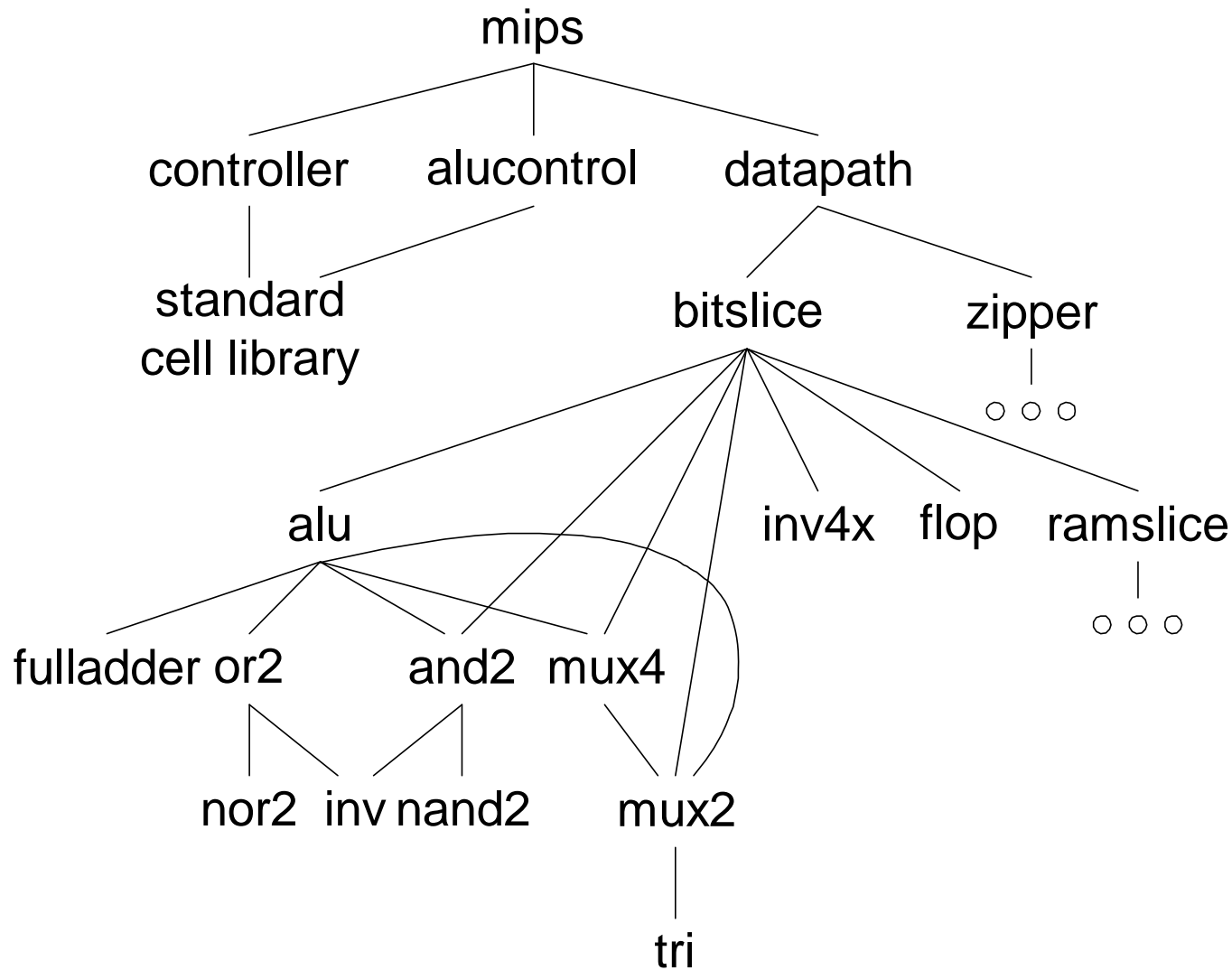


Design Partitioning : Structured Design

- **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: Flow

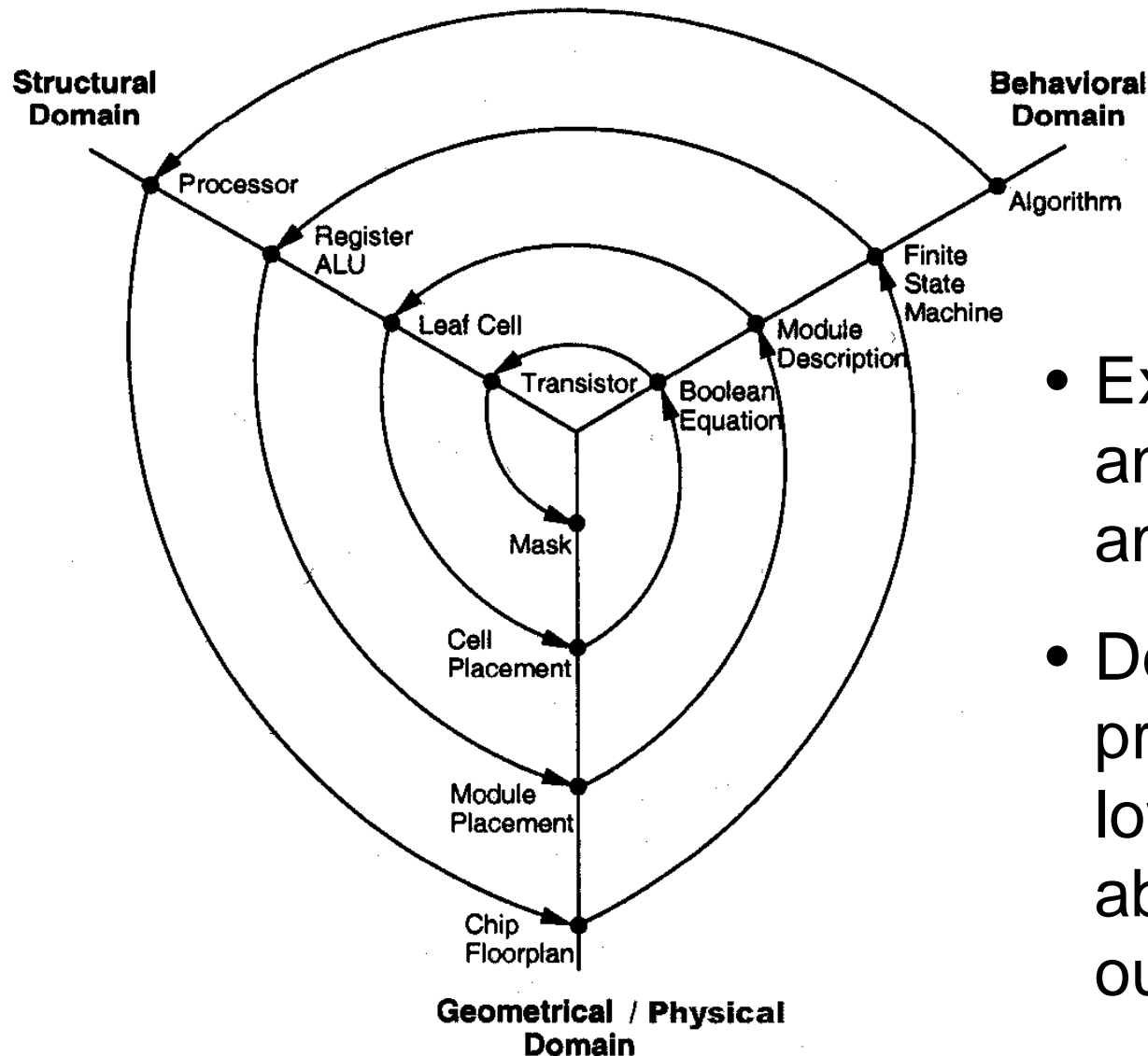


Design Partitioning : Y-Chart

- **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.



Hardware Description Language (HDL)

- Hardware Description Languages
 - Widely used in logic design
 - Verilog
 - Very-High-Speed-Integrated-Circuit HDL (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);
```

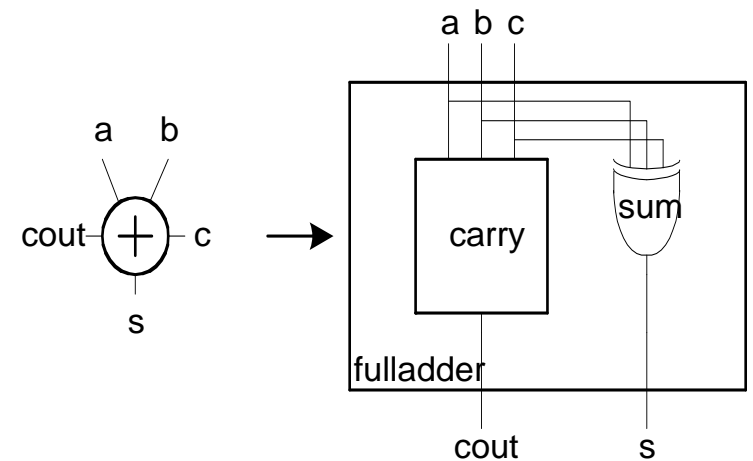
```
endmodule
```

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

```
assign cout
```

```
    = (a&b) | (a&c) | (b&c);
```

```
endmodule
```



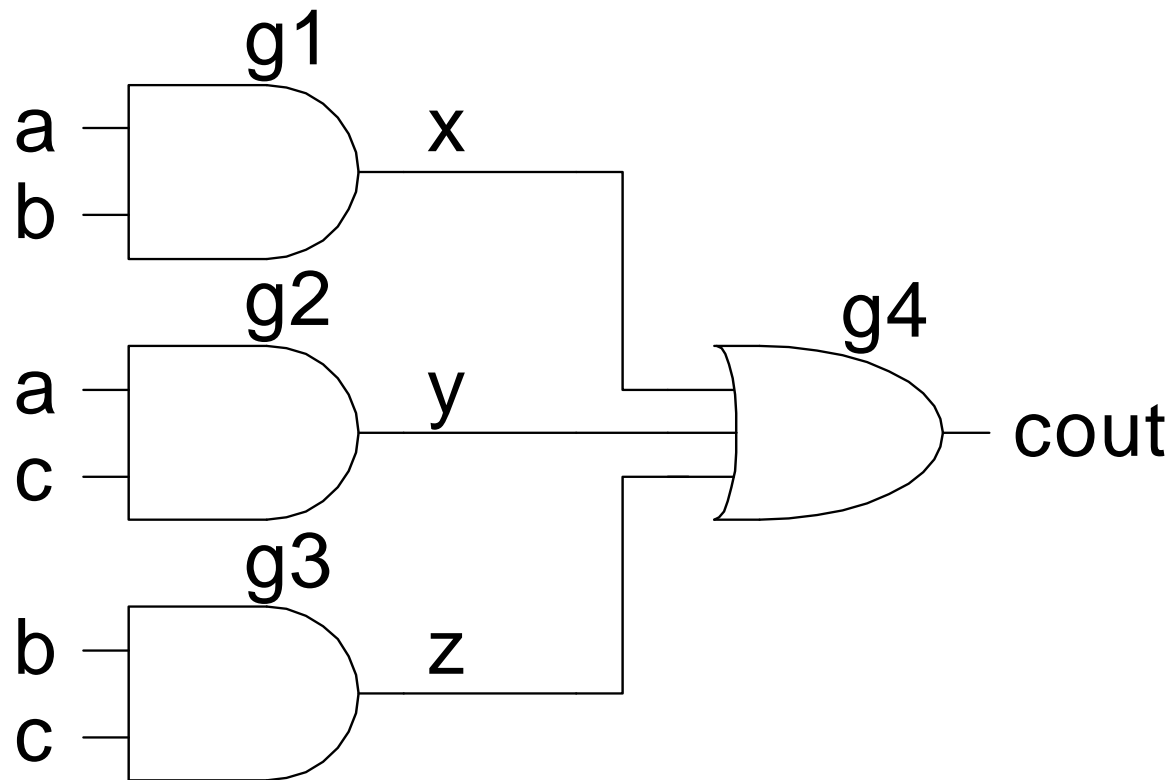
Circuit Design

- 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

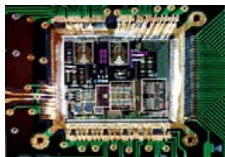


Carry Logic Example: Logic Level

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

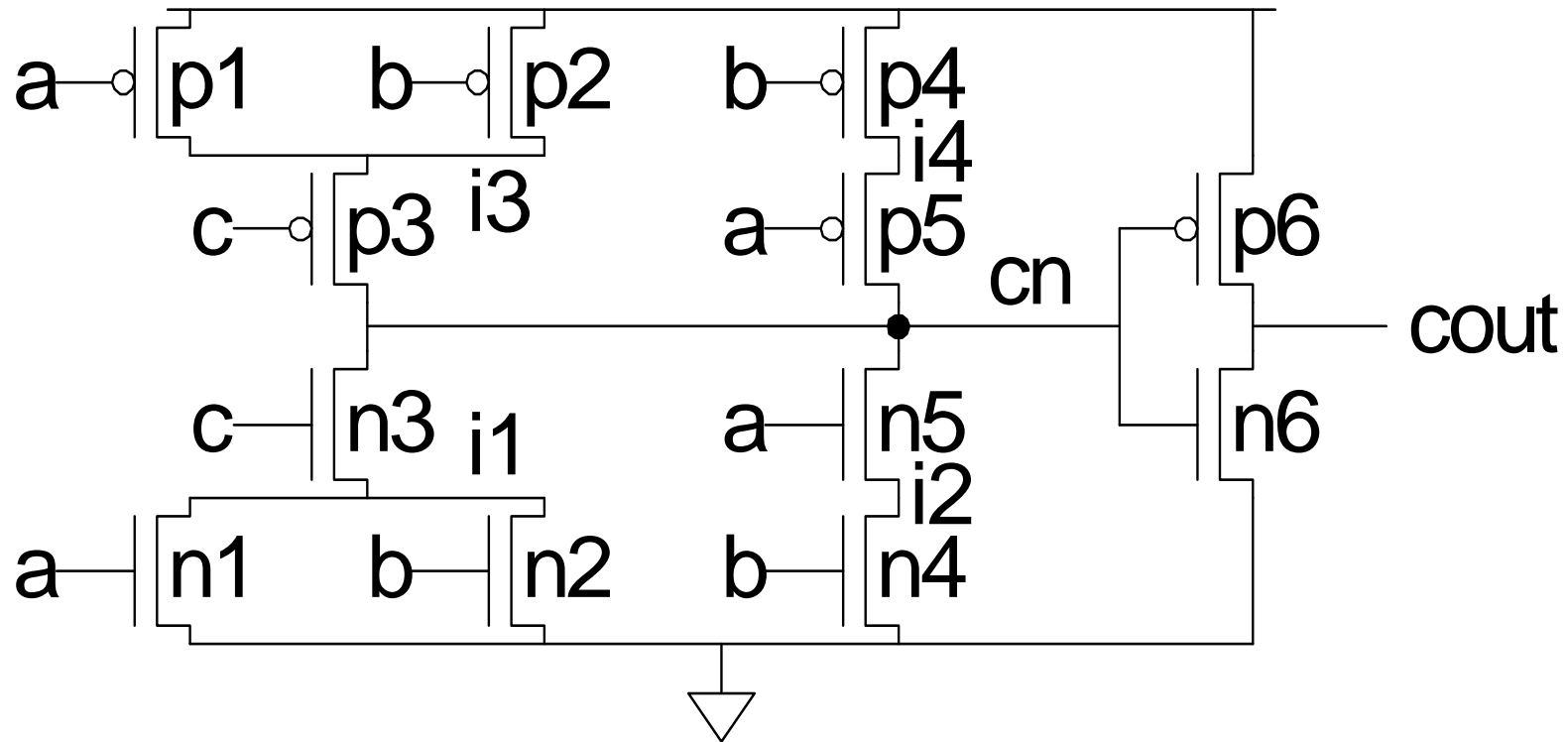


Transistors? Gate Delays?



Carry Logic Example: Transistor Level

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



Transistors? Gate Delays?

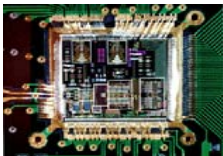
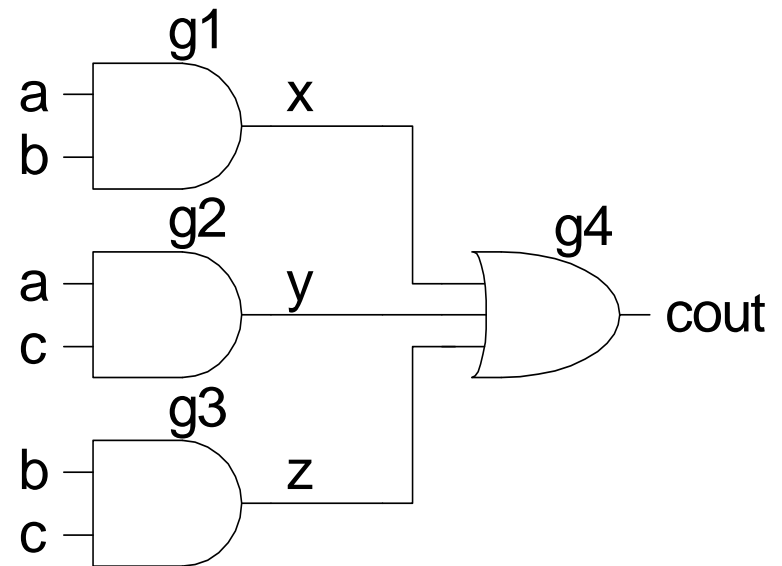


Carry Logic Example: Gate-level Netlist

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

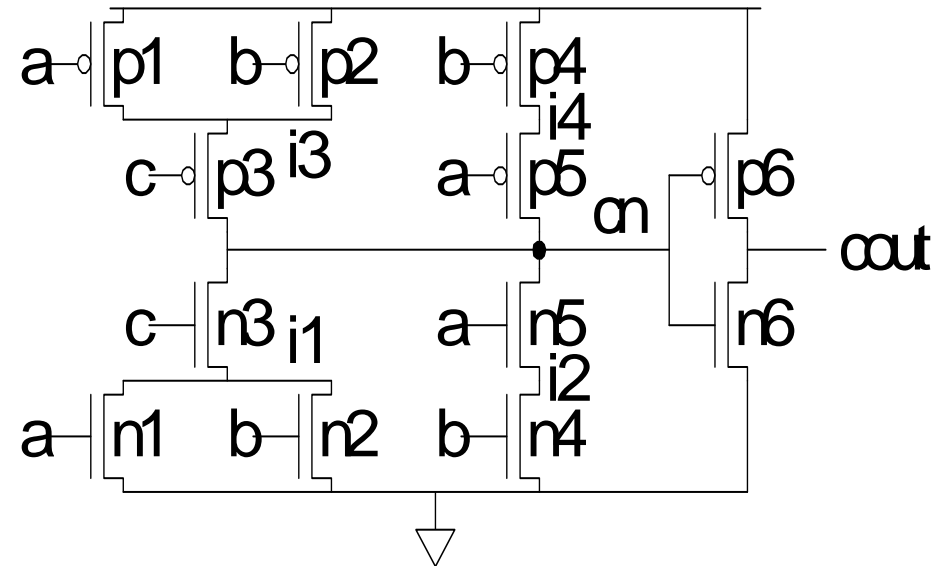
```
    wire  x, y, z;  
    and g1(x, a, b);  
    and g2(y, a, c);  
    and g3(z, b, c);  
    or  g4(cout, x, y, z);
```

```
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.



Layout Design Rules

- Interface between designer and process engineer
- Guidelines for constructing process masks
- Unit dimension: Minimum line width
 - scalable design rules: lambda parameter
 - absolute dimensions (micron rules)



Layout Design Rules

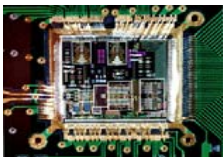
- Chips are specified with set of masks
- Minimum dimensions of masks determine transistor size (and hence speed, cost, and power)
- Feature size f = distance between source and drain
 - Set by minimum width of polysilicon
- Normalize for feature size when describing design rules
- Express rules in terms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



Layout Design Rules : Simplified Form

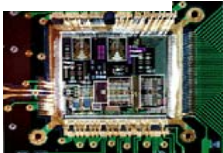
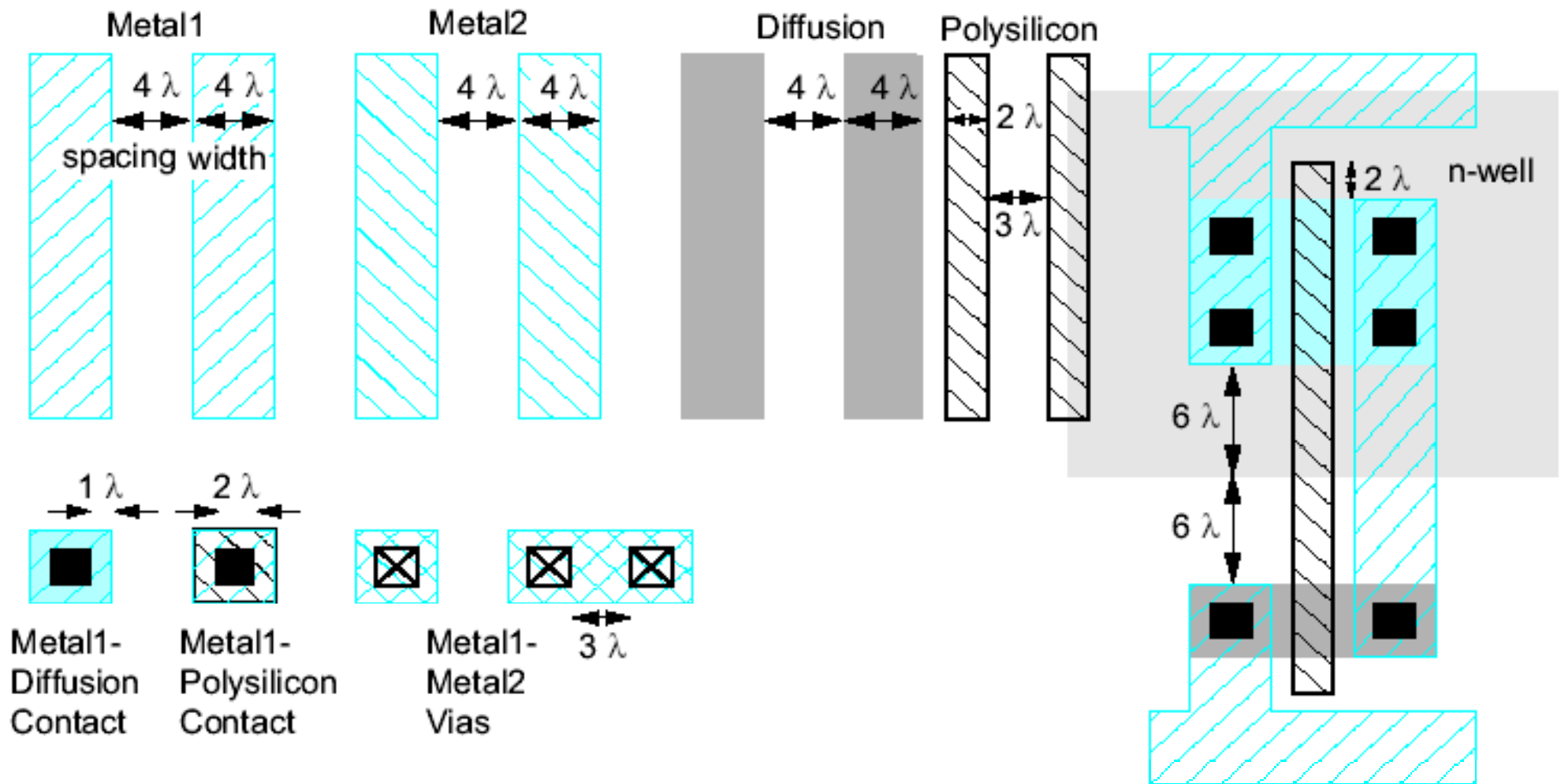
- Metal and diffusion have minimum width and spacing of 4λ .
- Contacts are $2\lambda \times 2\lambda$ and surrounded by 1λ layer.
- Polysilicon width is 2λ .
- Polysilicon overlaps diffusion by 2λ or 1λ depending on situation.
- Polysilicon and contacts have spacing 3λ .
- n-well surrounds PMOS transistors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

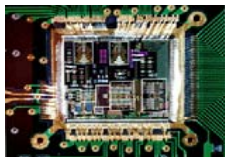
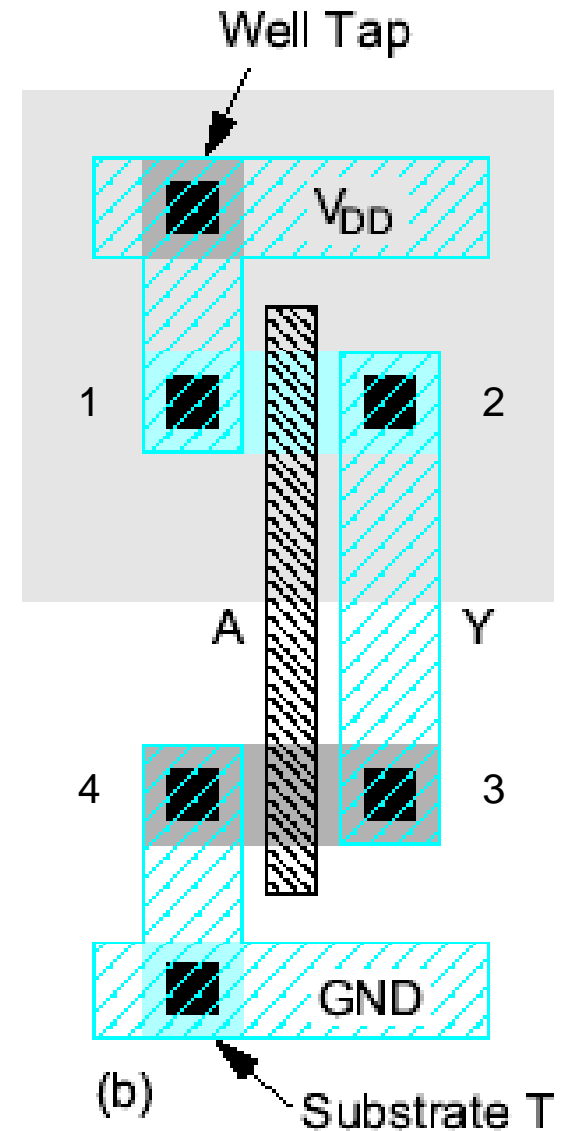
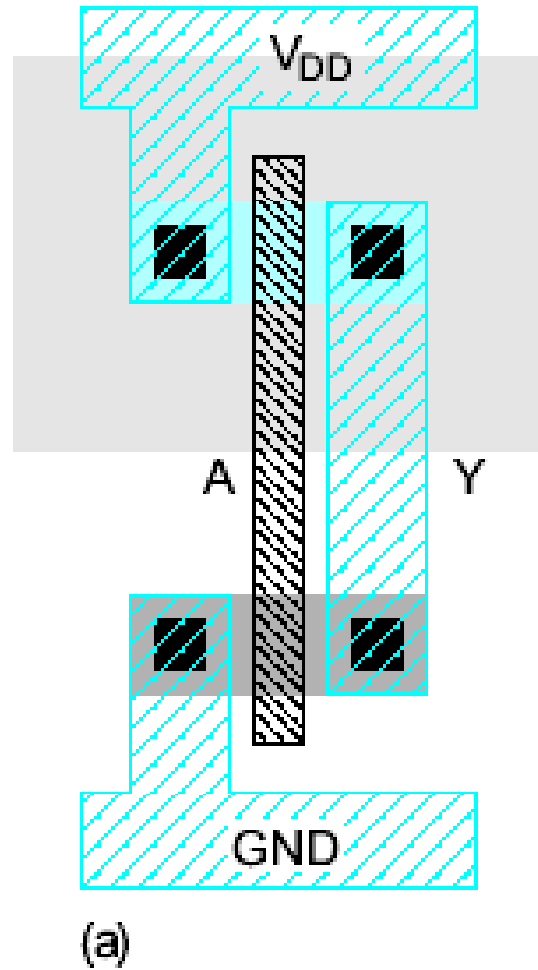
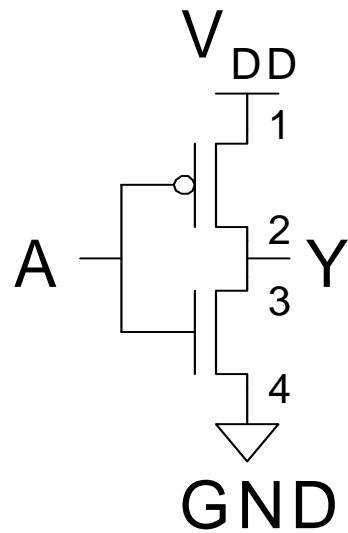


Layout Design Rules : Simplified Form










- Conservative rules to get you started



Logic Gate Layout : Inverter





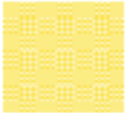












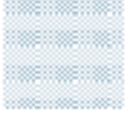


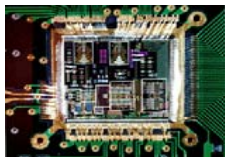
CMOS Process Layers

Layer	Color	Representation
Well (p,n)	Yellow	
Active Area (n+,p+)	Green	
Select (p+,n+)	Green	
Polysilicon	Red	
Metal1	Blue	
Metal2	Magenta	
Contact To Poly	Black	
Contact To Diffusion	Black	
Via	Black	

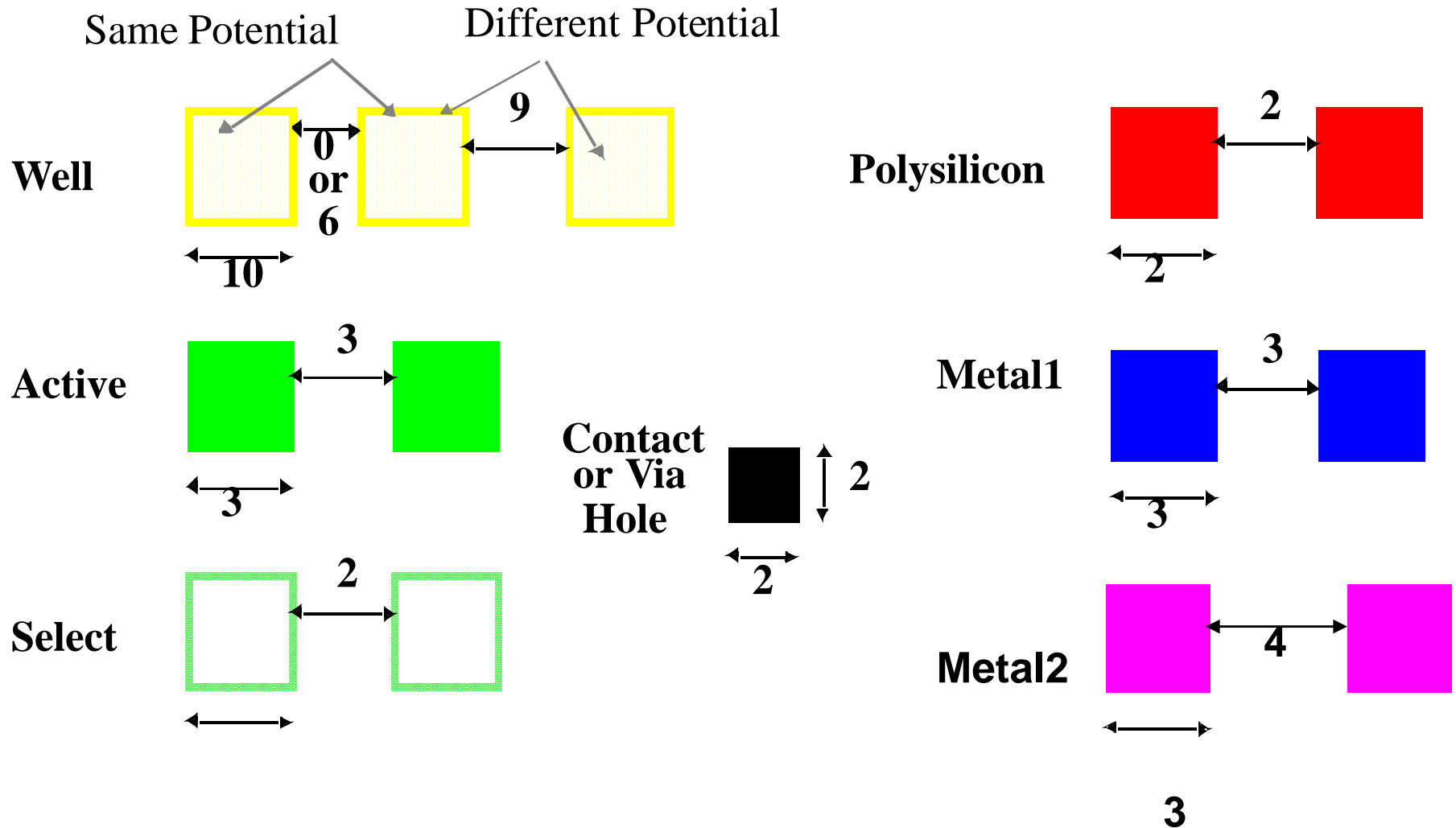


Layers in 0.25 μm CMOS process

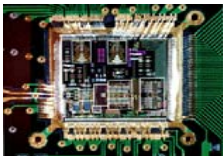
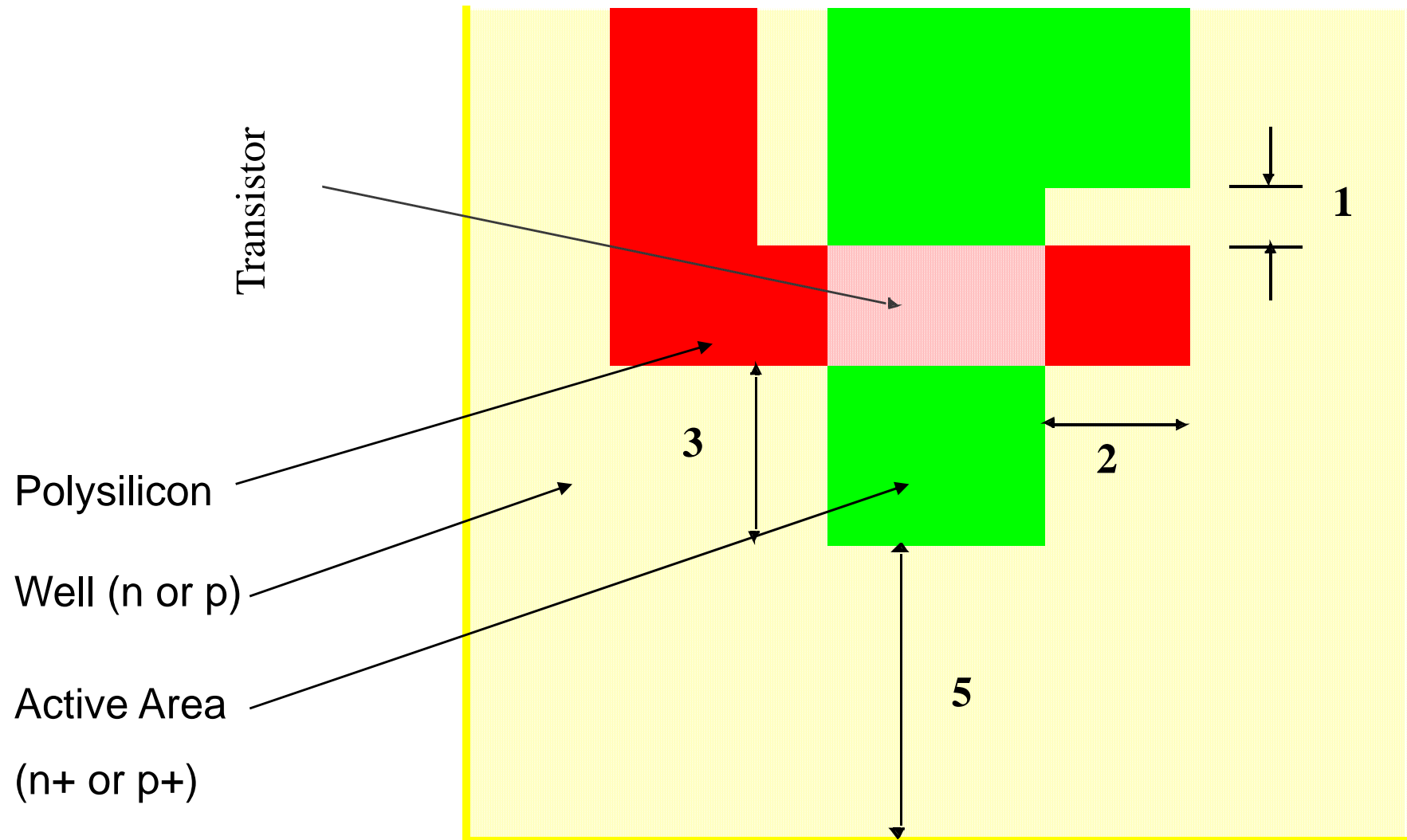
Layer Description	Representation				
metal					
	m1	m2	m3	m4	m5
					
	nw				
					
polysilicon	poly				
contacts & vias					
	ct	v12,v23,v34,v45	nwc	pwc	
					
	ndif	pdif	nfet	pfet	
select					
	nplus	pplus	prb		



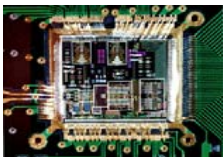
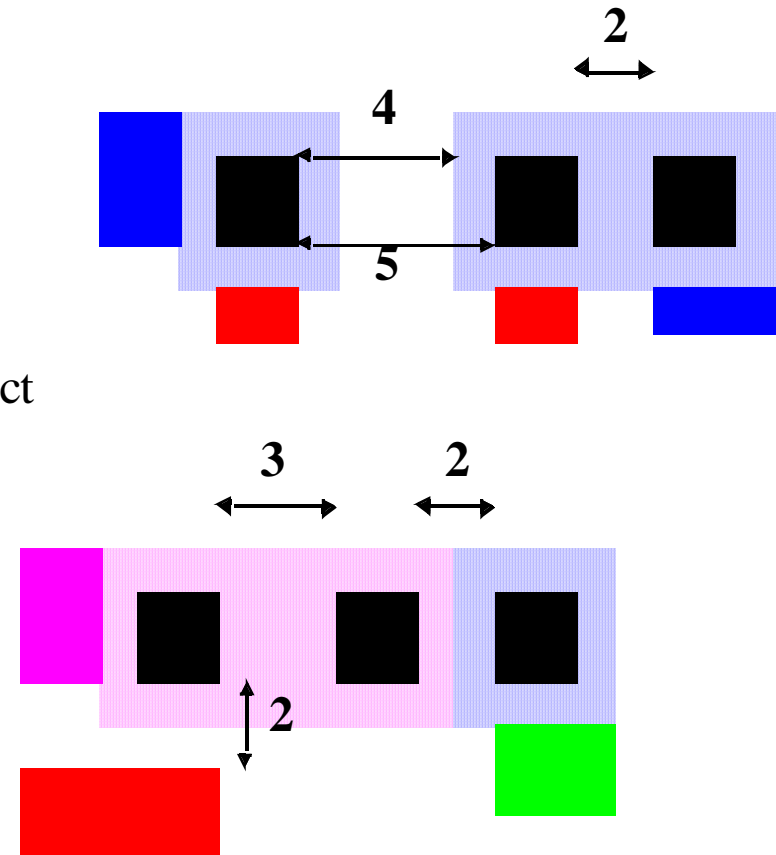
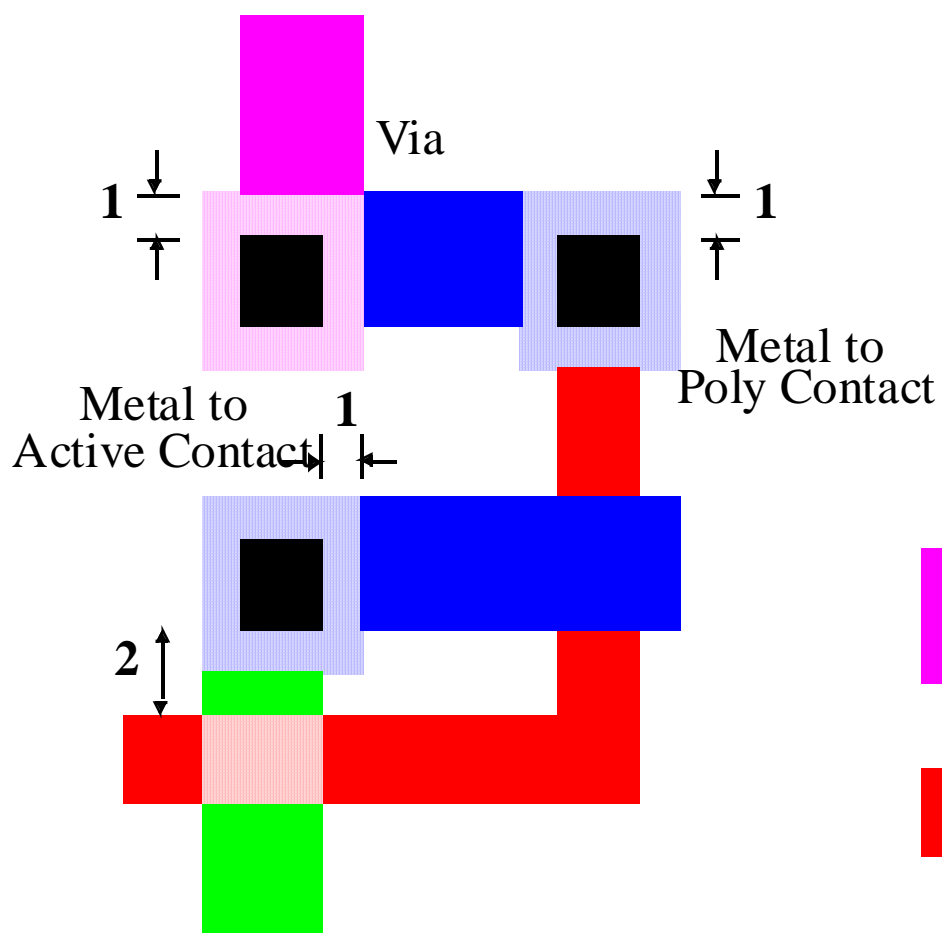
Intra-Layer Design Rules



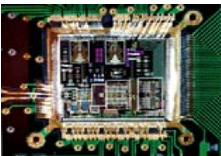
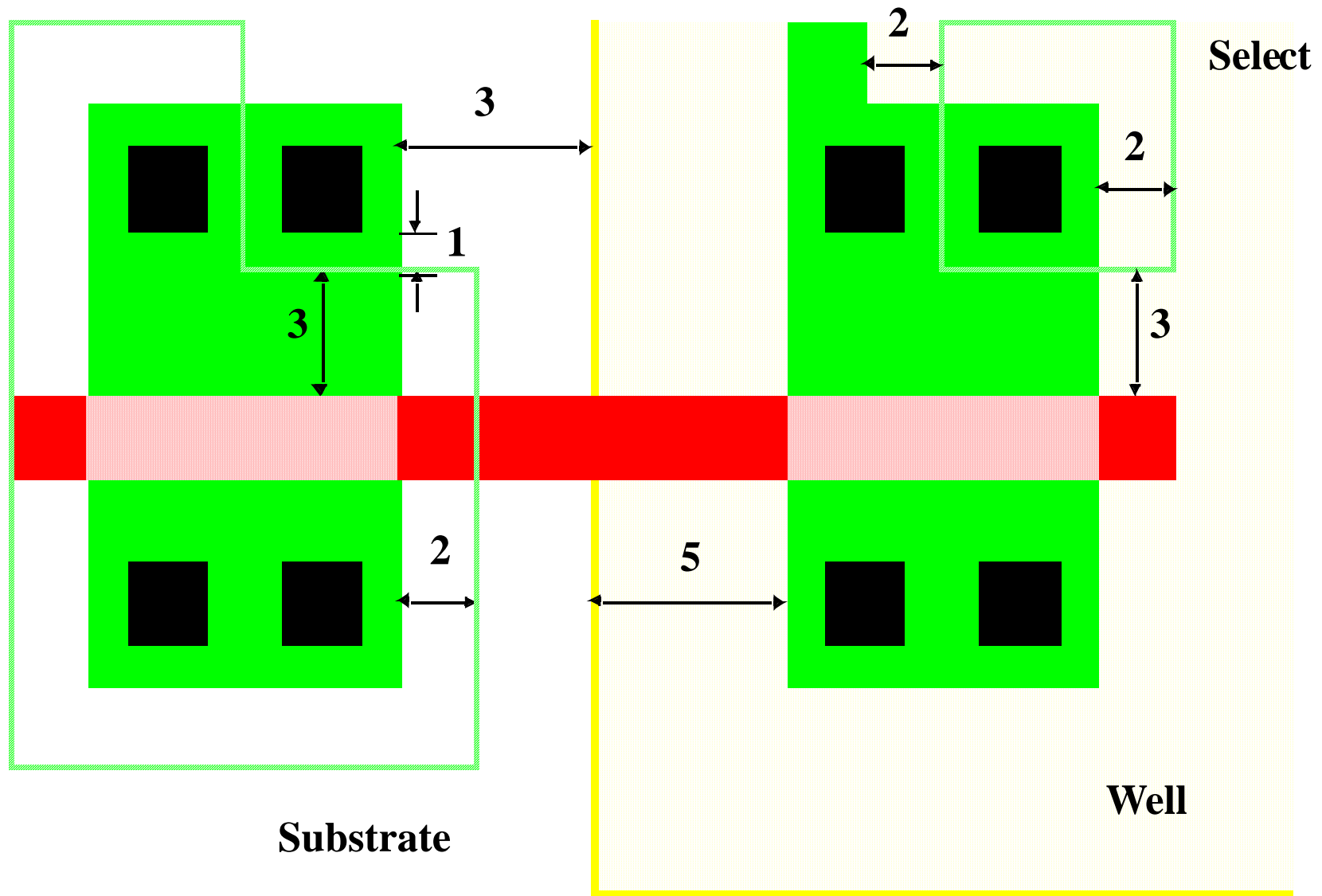
Transistor Layout



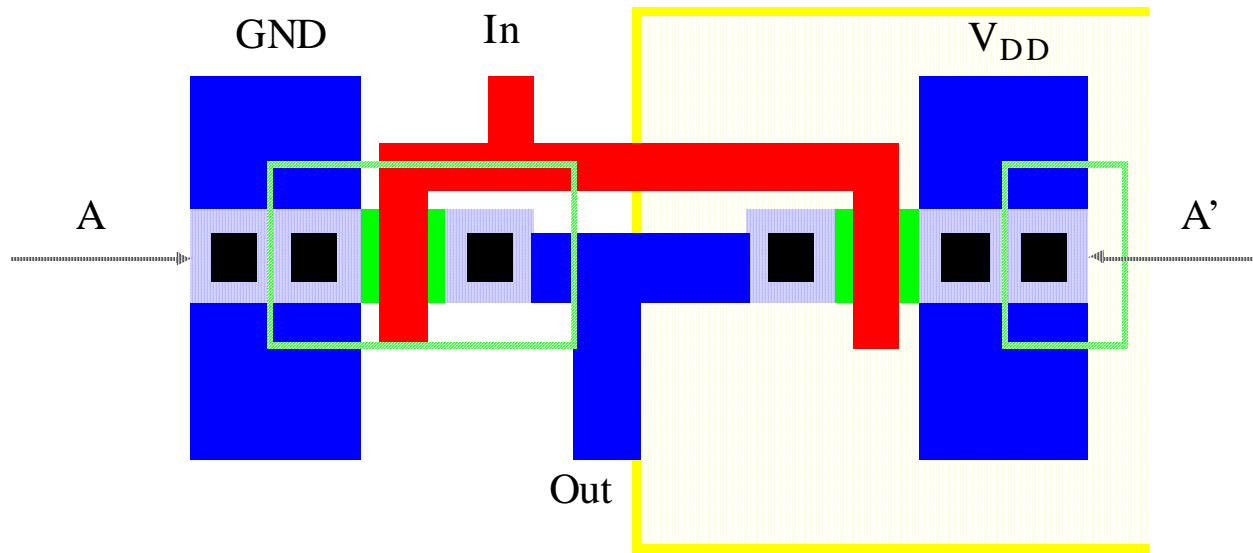
Vias and Contacts



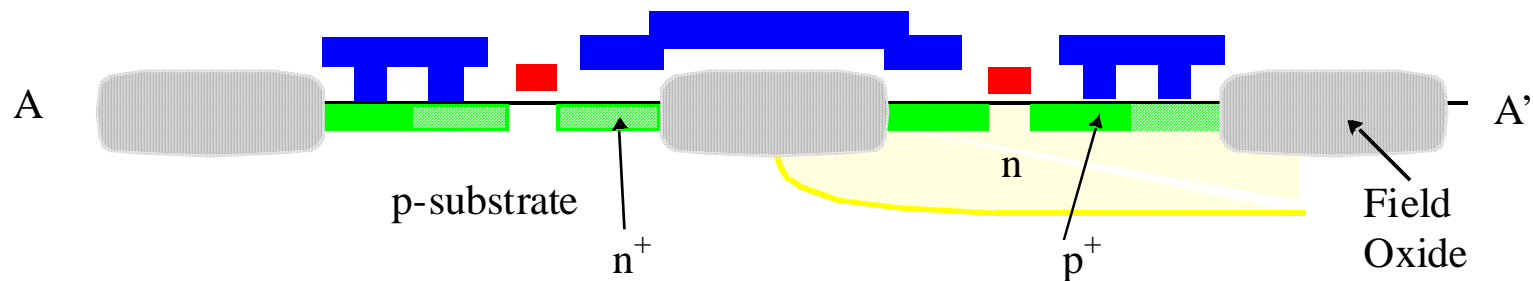
Select Layer



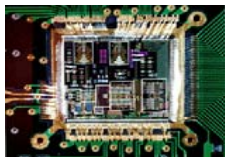
CMOS Inverter Layout



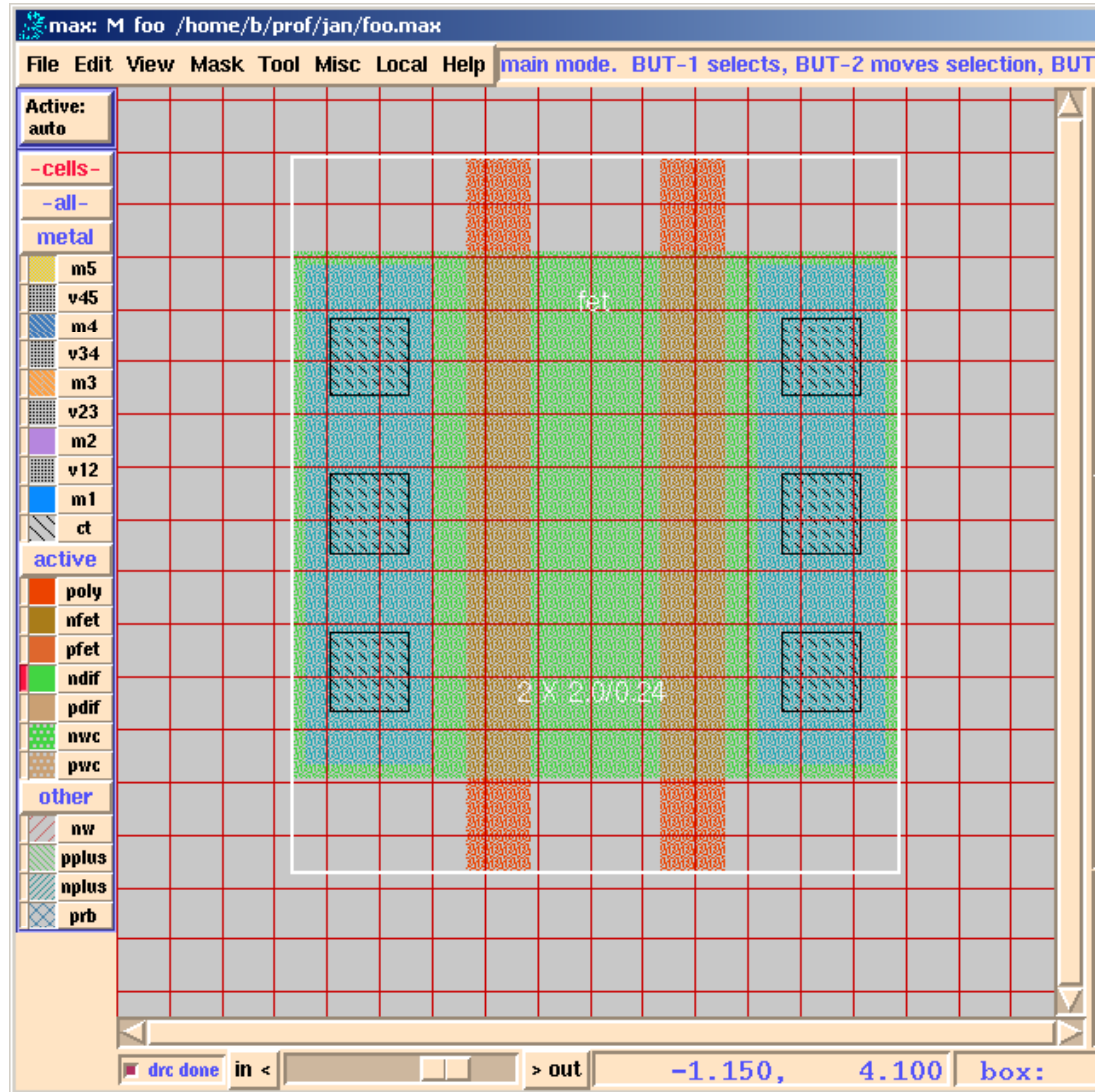
(a) Layout



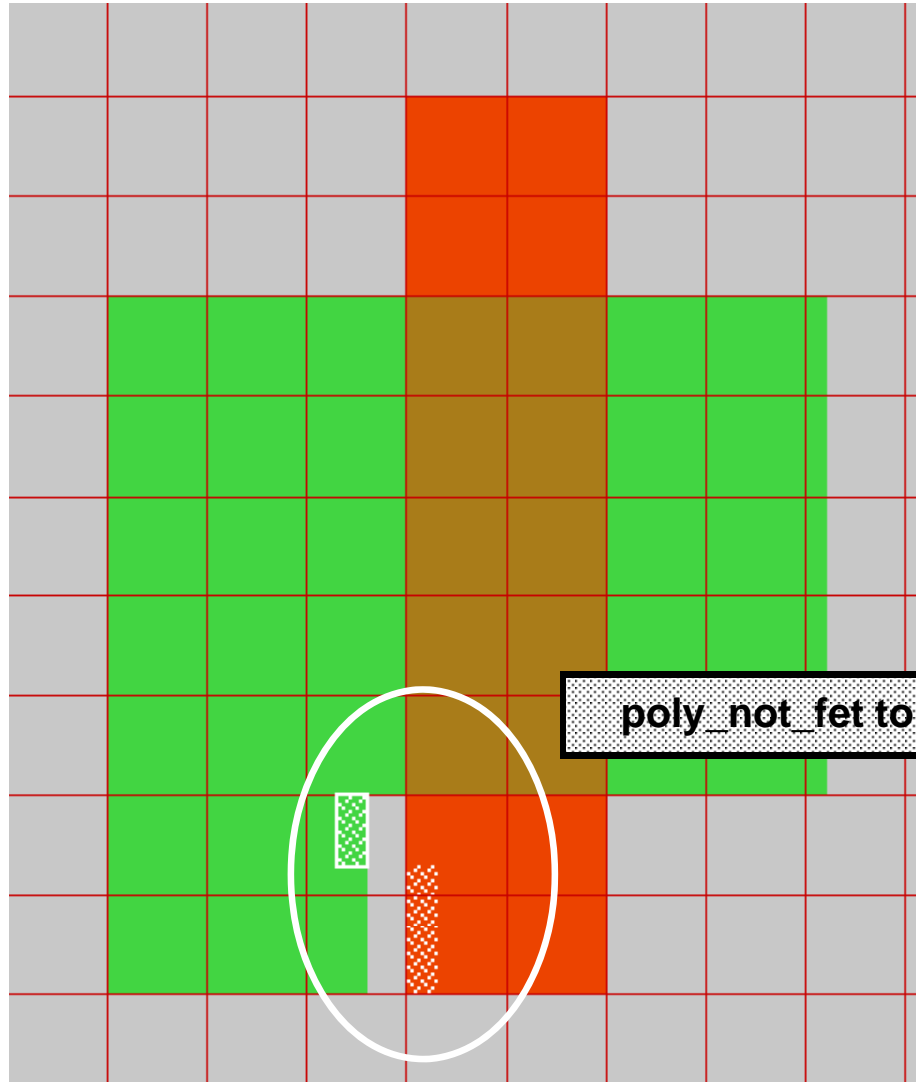
(b) Cross-Section along A-A'



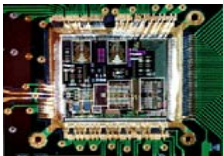
Layout Editor



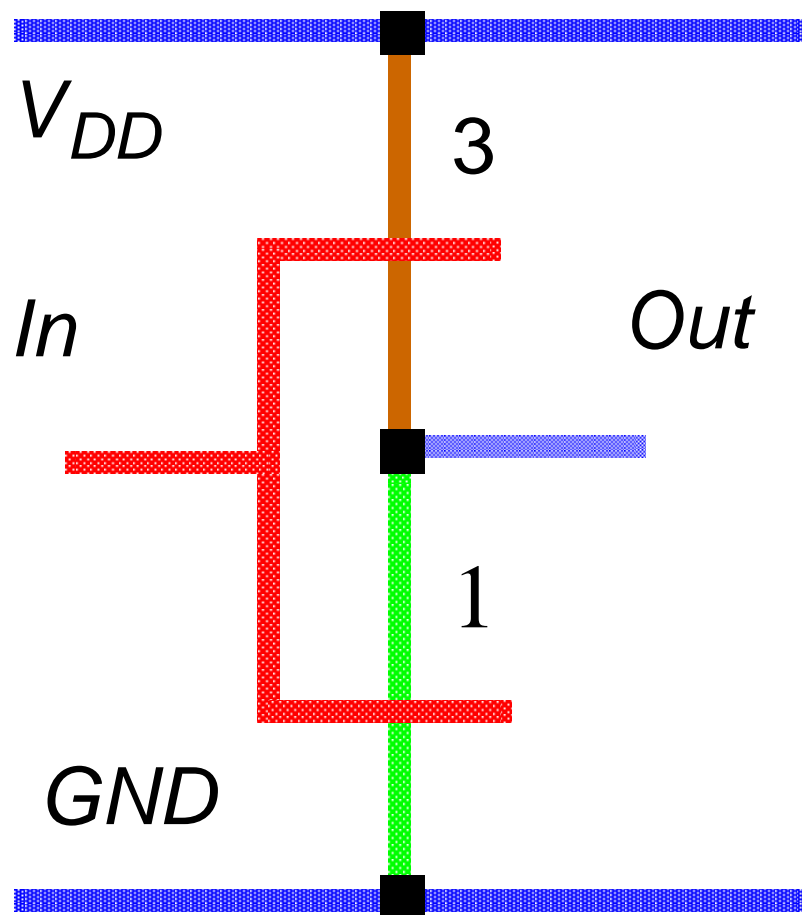
Design Rule Checker



poly_not_fet to all_diff minimum spacing = 0.14 um.

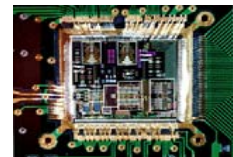


Sticks Diagram



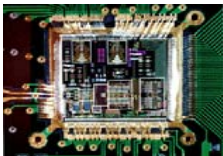
- Dimensionless layout entities
- Only topology is important
- Final layout generated by “compaction” program

Stick diagram of inverter

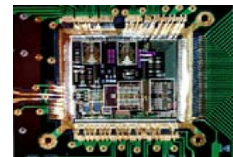
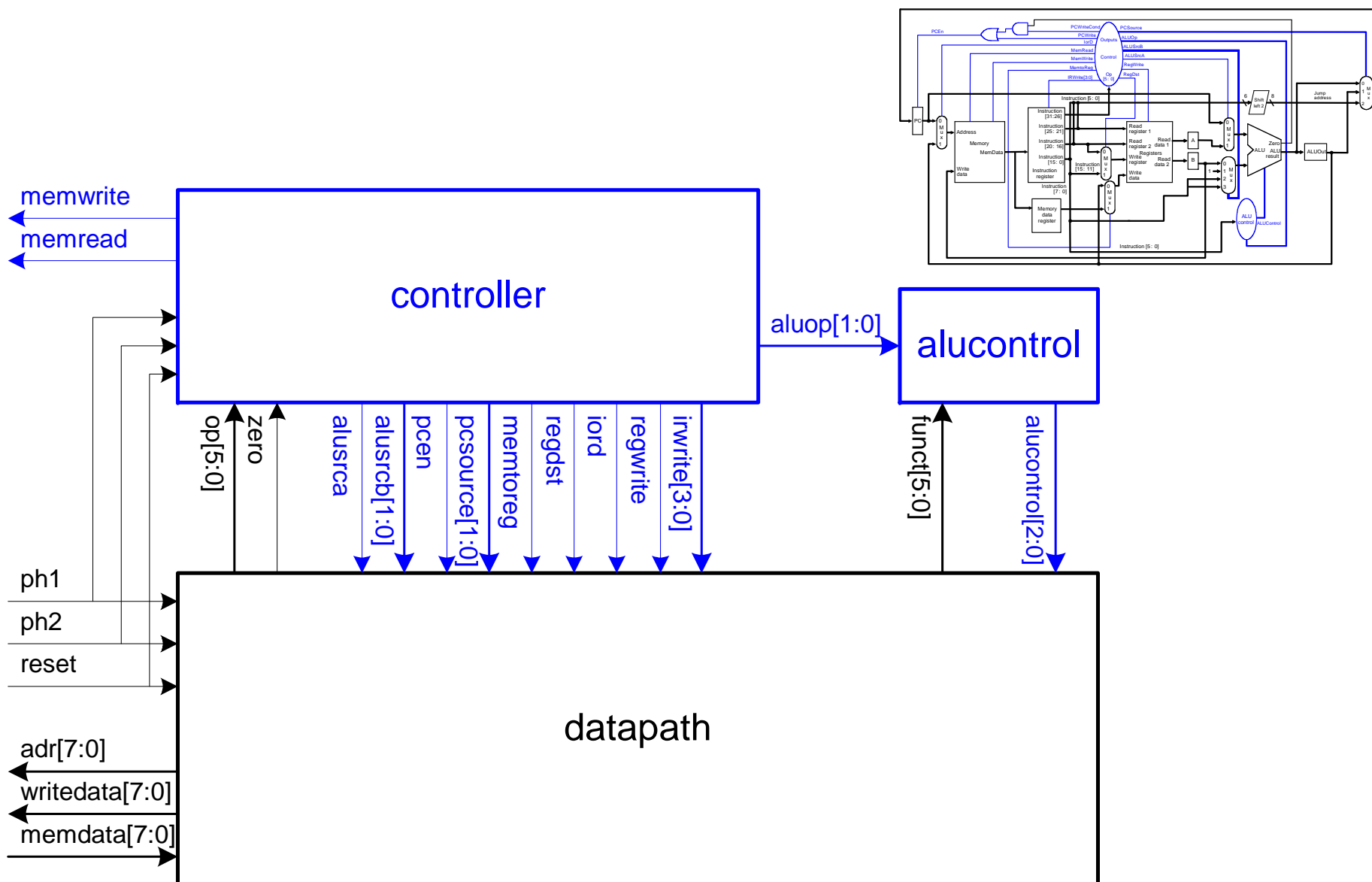


Logic Gate Layout

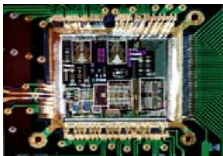
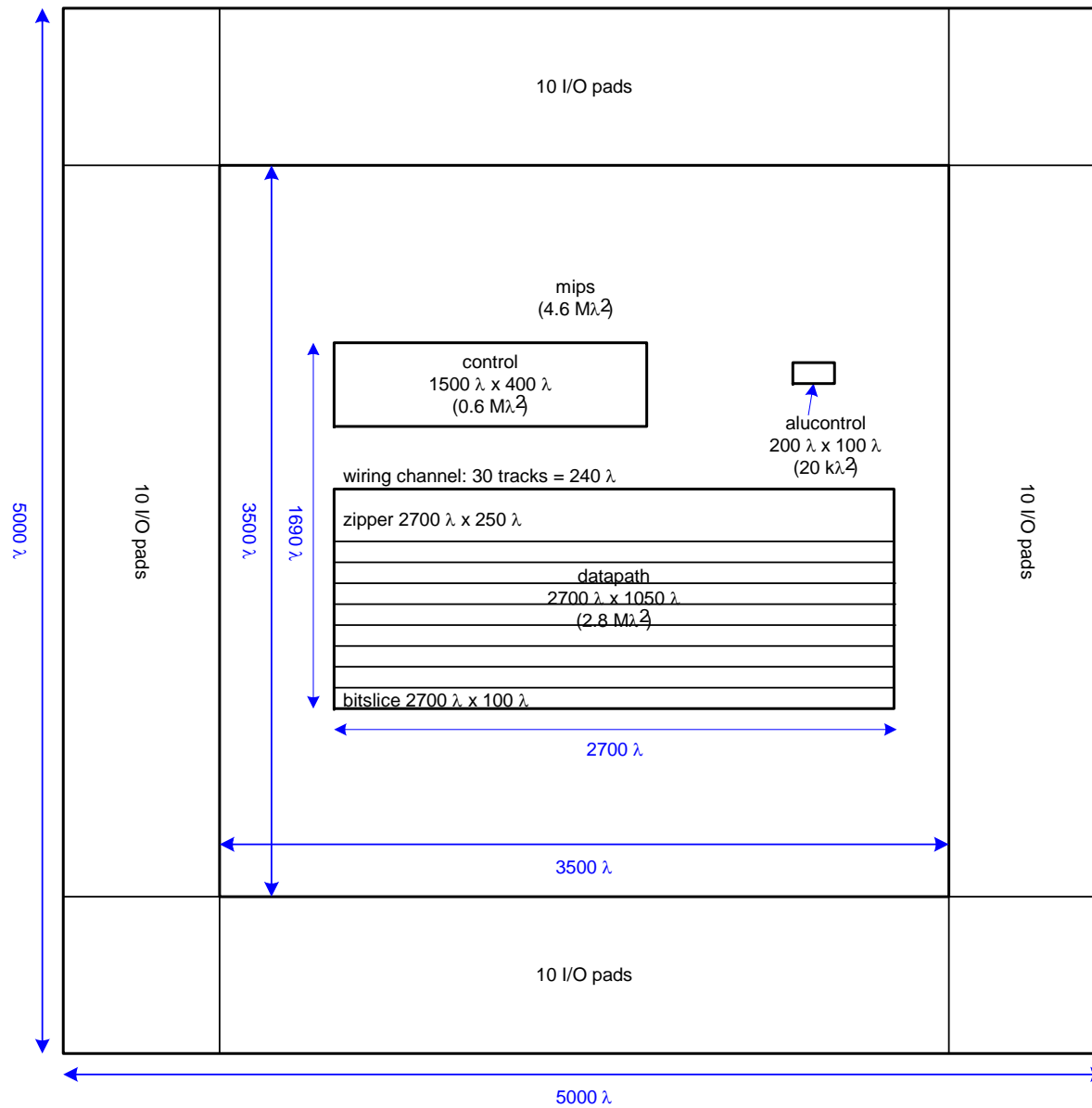
- Layout can be very time consuming
 - Design gates to fit together nicely
 - Build a library of standard cells
- Standard cell design methodology
 - V_{DD} and GND should abut (standard height)
 - Adjacent gates should satisfy design rules
 - nMOS at bottom and pMOS at top
 - All gates include well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



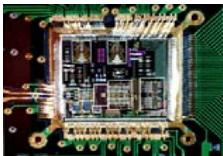
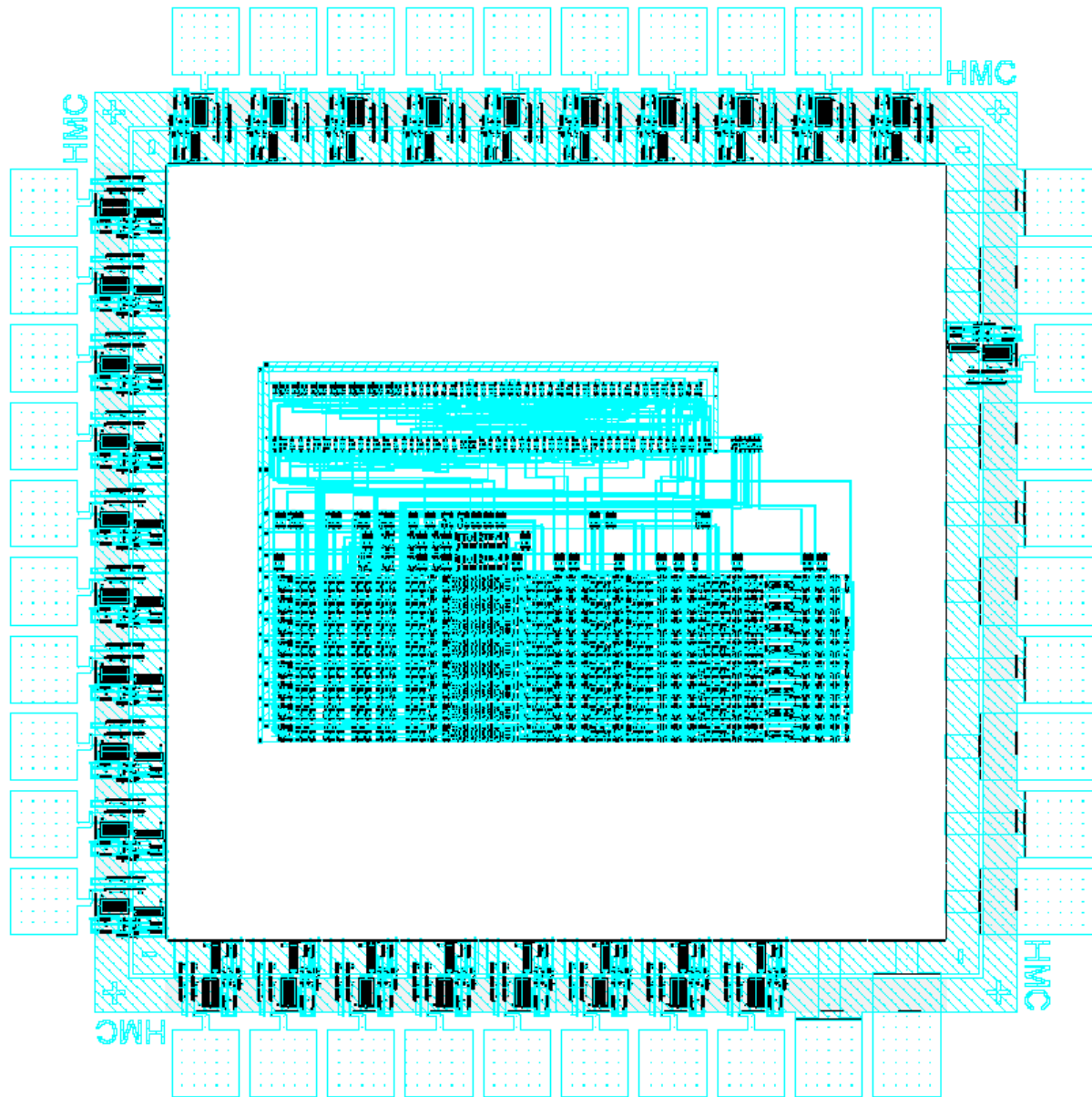
MIPS: Block Diagram



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)	$1000 - 1500 \lambda^2 / \text{transistor}$
datapath	$250 - 750 \lambda^2 / \text{transistor}$ or $6 \text{ WL} + 360 \lambda^2 / \text{transistor}$
SRAM	$1000 \lambda^2 / \text{bit}$
DRAM (in a DRAM process)	$100 \lambda^2 / \text{bit}$
ROM	$100 \lambda^2 / \text{bit}$



Design Verification

- Fabrication is slow & expensive
 - MOSIS 0.6 μ m: \$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!

