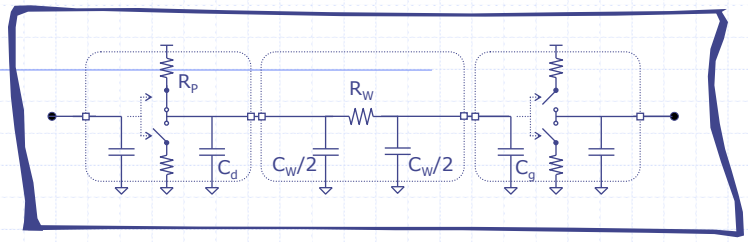
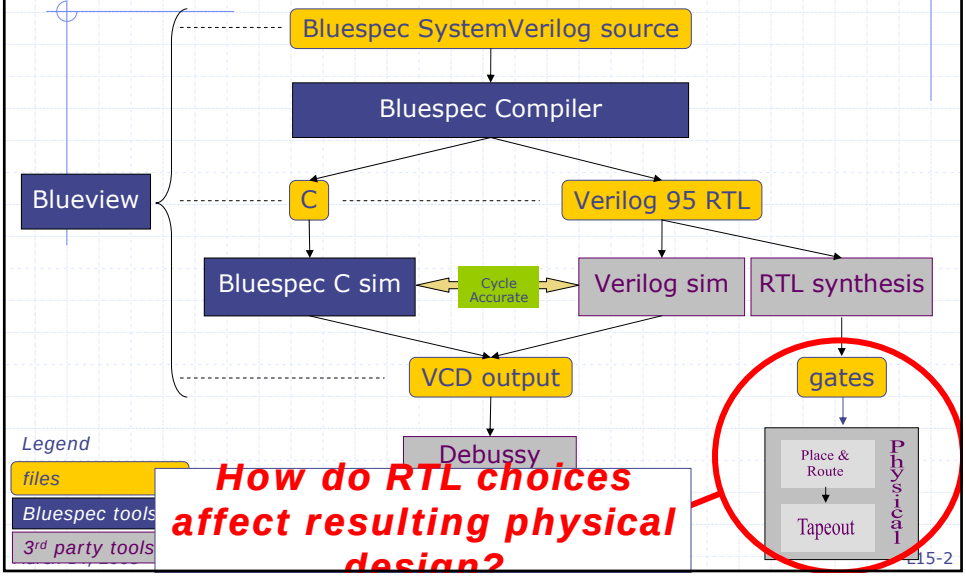
Physical Design - 1



Arvind
Computer Science & Artificial Intelligence Lab
Massachusetts Institute of Technology

6.375 Standard Cell Design Flow



Metrics for Chip "Quality"

- Area
  - Size affects manufacturing and packaging costs
- Performance
  - Does chip meet market performance goals?
- Power
  - Peak power affects packaging cost (current supply, heat removal)
  - Energy usage affects battery life

Iron Law of Performance

$$\text{Performance} = \frac{\text{Operations}}{\text{Clock Cycle}} \times \frac{\text{Clock Cycles}}{\text{Second}}$$

Concurrency in RTL Design

Clock Frequency of Physical Design

These are not independent parameters!

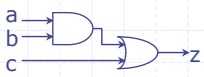
Clock frequency set by delay of circuit components in critical path

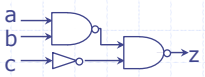
What is synthesis ?

- Synthesis tools (e.g., Design Compiler) coverts RTL into gate level netlist given a gate library
  - infer logic and state elements
    - Rather straightforward unless the language semantics complicate it
  - perform technology-independent optimizations
    - logic simplification, state assignment, …
  - map elements to the target technology
  - perform technology-dependent optimizations
    - multi-level logic optimization, choose gate strengths to achieve speed goals, …

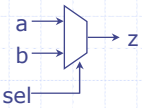
Logic Synthesis

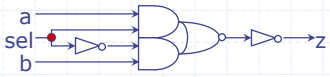
```
assign z = (a & b) | c;
```





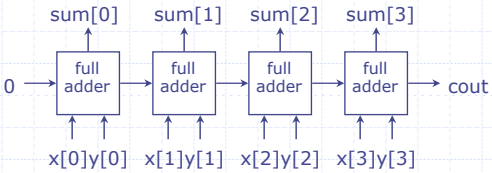

```
// dataflow
assign z = sel ? a : b;
```






```
wire [3:0] x,y,sum;
wire cout;
assign {cout,sum} = x + y;
```

As a default + is implemented as a ripple carry editor



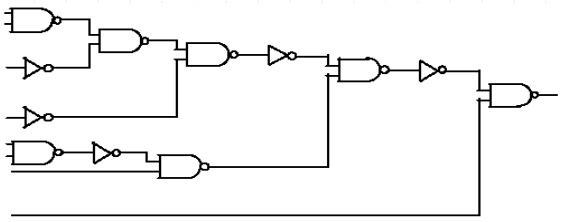
Technology-independent optimizations

- Two-level boolean minimization
  - Quine-McCluskey
- Optimizing finite state machines
  - look for an equivalent FSM that has fewer states
  - Choose an FSM state encodings that minimizes the size of state storage + size of logic to implement next state and output functions).

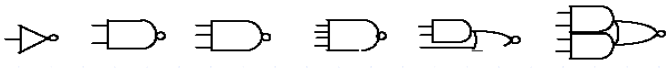
None of these operations is completely isolated from the target technology. But experience has shown that it's advantageous to reduce the size of the problem as much as possible before starting the technology-dependent optimizations

Mapping to target technology

Problem statement: find an "optimal" mapping of this circuit:

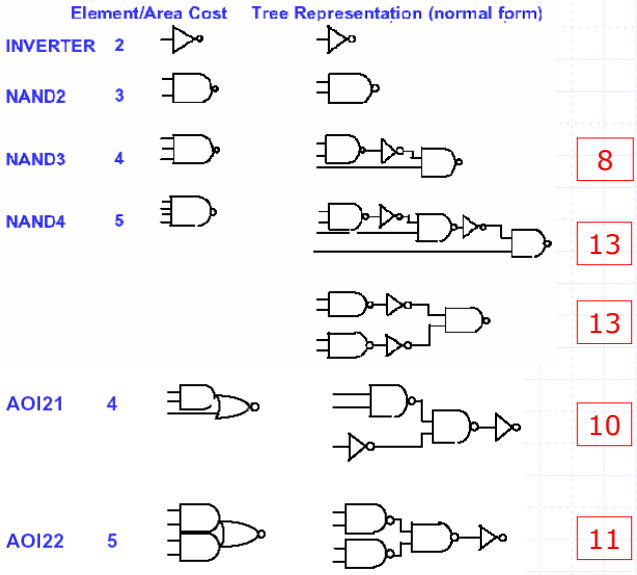


Into this library:

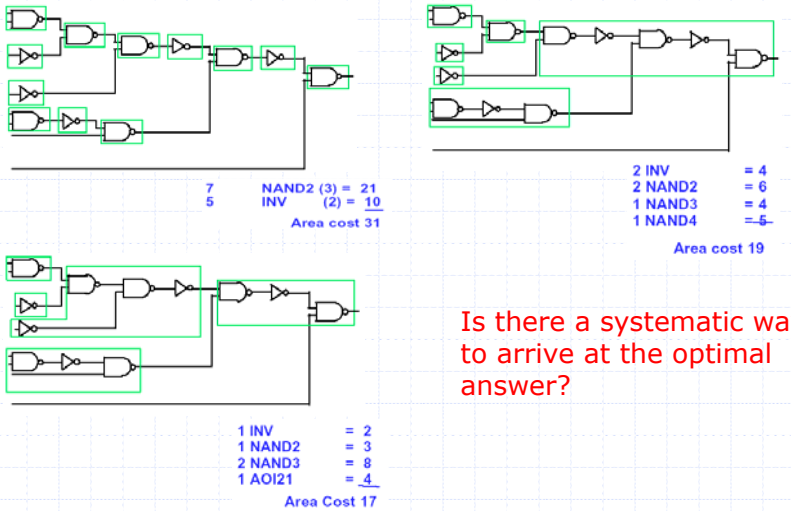


Popular approach: DAG covering (K. Keutzer)

A library of gates



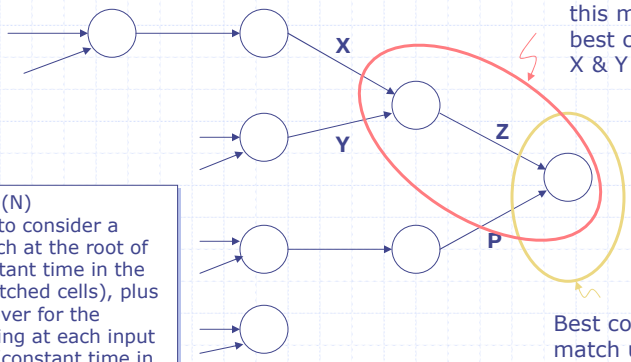
Possible implementations



Is there a systematic way to arrive at the optimal answer?

Use dynamic programming!

Optimal cover for a tree consists of a best match at the root of the tree plus the optimal cover for the sub-trees starting at each input of the match.

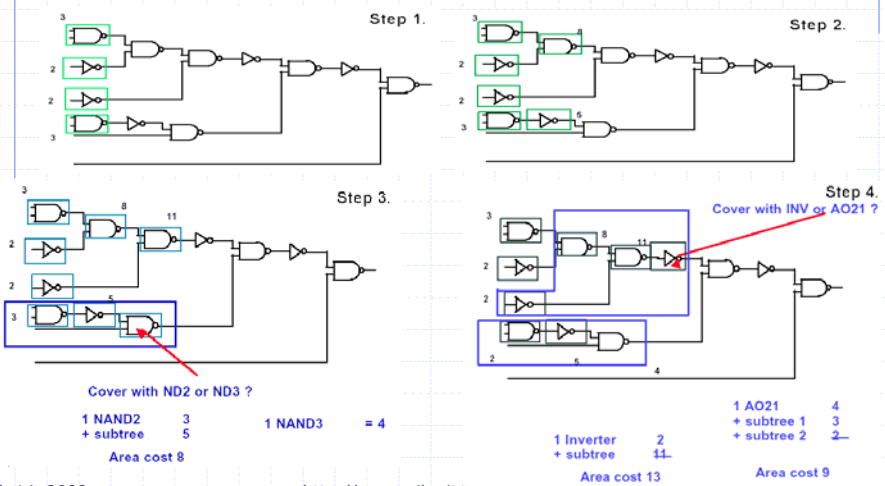


Best cover for this match uses best covers for P, X & Y

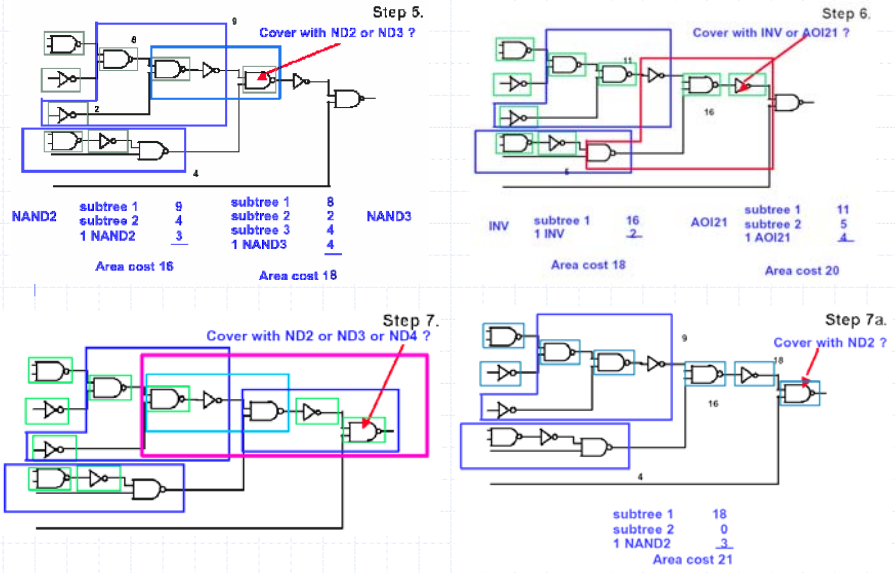
Best cover for this match uses best covers for P & Z

Complexity: O(N)
we only need to consider a best-cost match at the root of the tree (constant time in the number of matched cells), plus the optimal cover for the subtrees starting at each input to the match (constant time in the fanin of each match)

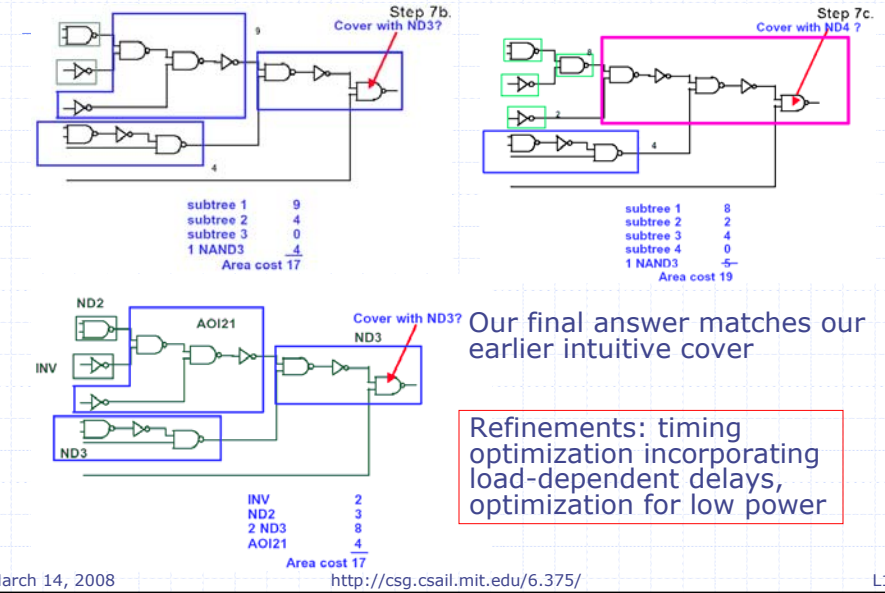
Optimal tree covering example



Example *cont.*



Example *cont.*



Our final answer matches our earlier intuitive cover

Refinements: timing optimization incorporating load-dependent delays, optimization for low power

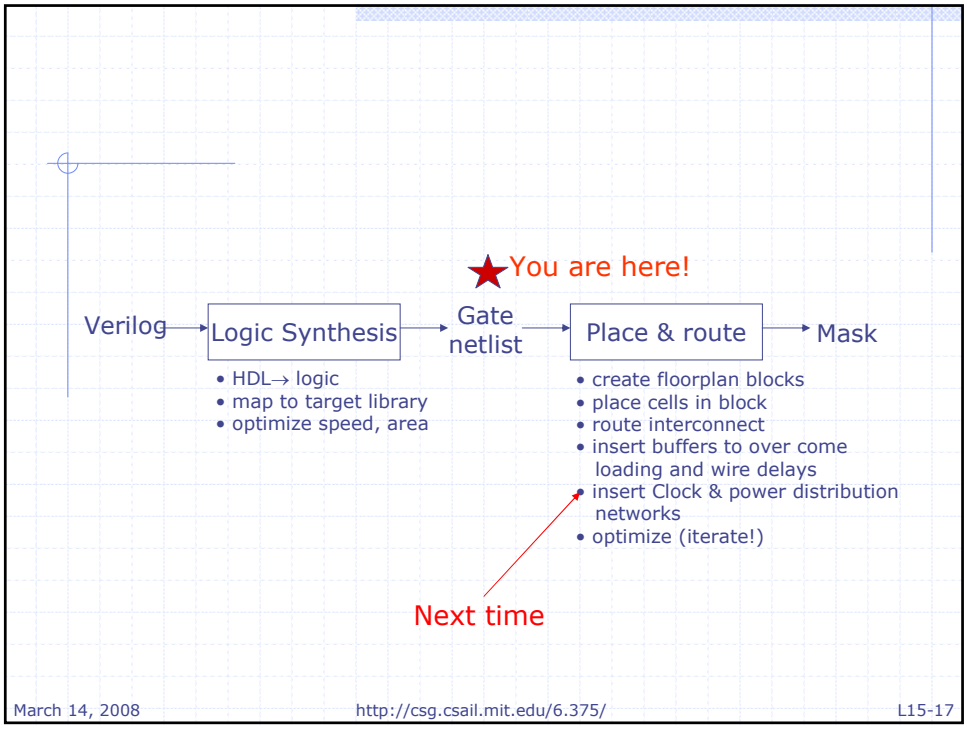
DAG Covering

2-input NAND gates + inverters

- Represent input netlist in *normal form* ("subject DAG")
- Represent each library gate in normal form ("primitive DAGs").
- Goal: find a minimum cost covering of the subject DAG by the primitive DAGs.
  - If the subject and primitive DAGs are *trees*, use dynamic programming for finding the optimum cover
  - Partition subject DAG into a forest of trees (each gate with fanout > 1 becomes root of a new tree), generate optimal solutions for each tree, stitch solutions together

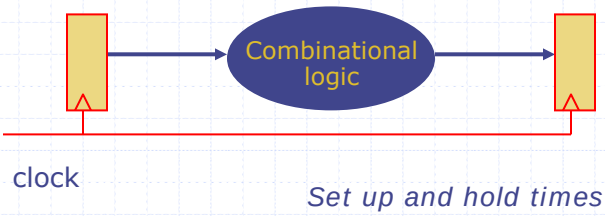
Technology-dependent optimizations

- Additional library components: more complex cells may be slower but will reduce area for logic off the critical path.
- Load buffering: adding buffers/inverters to improve load-induced delays along the critical path
- Resizing: Resize transistors in gates along critical path
- Retiming: change placement of latches/registers to minimize overall cycle time
- Increase routability over/through cells: reduce routing congestion.



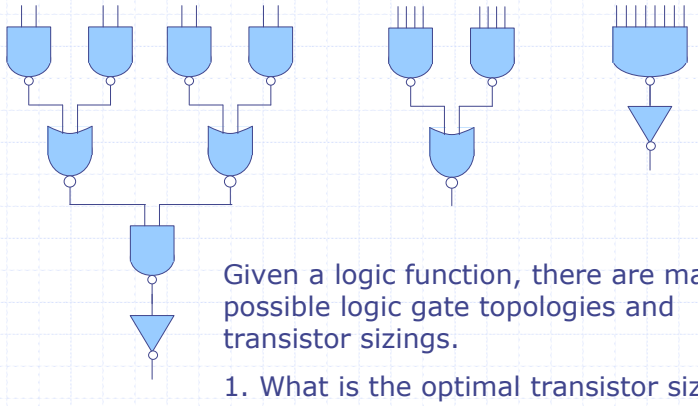
What determines clock cycle

- Fan-in of gates
- Fan-out of gates
- Wire lengths



Set up and hold times …

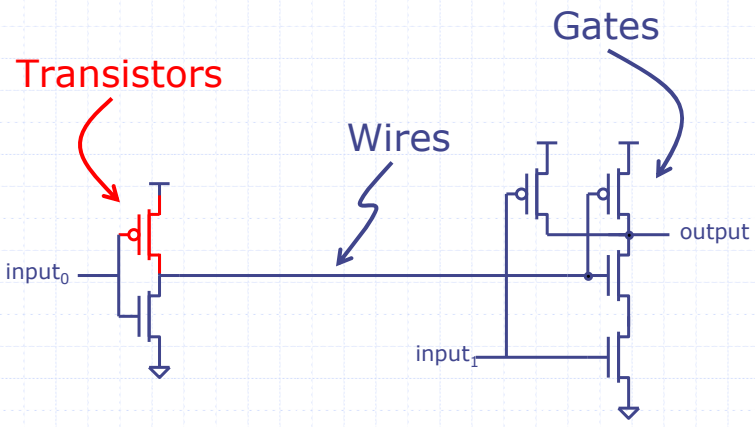
Which gate topology and transistor sizing is optimal?



Given a logic function, there are many possible logic gate topologies and transistor sizings.

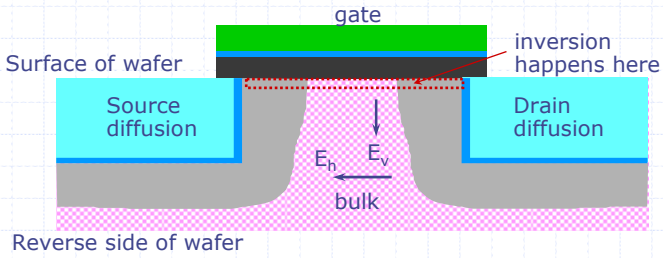
1. What is the optimal transistor sizing?
2. What is the optimal number of logic stages?

Basic CMOS Components



FET = Field-Effect Transistor

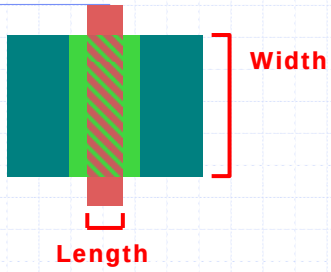
A four terminal device (gate, source, drain, bulk)

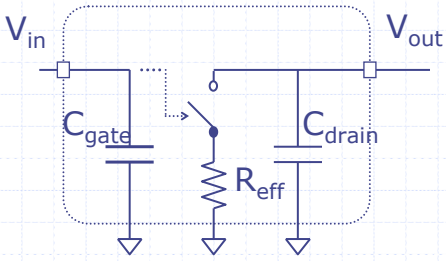


Inversion: A vertical field creates a channel between the source and drain.

Conduction: If a channel exists, a horizontal field causes a drift current from the drain to the source.

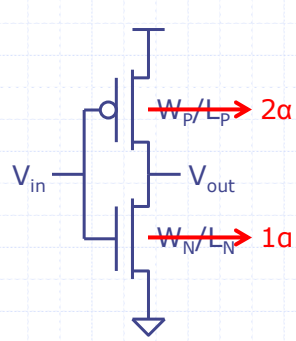
RC modeling of delay in MOSFET transistors

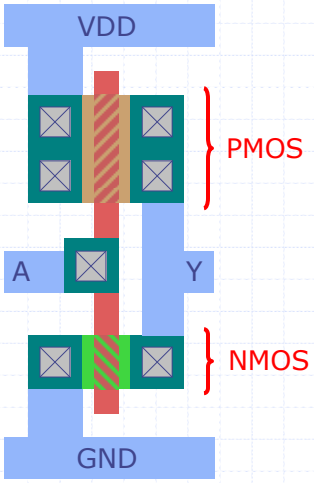




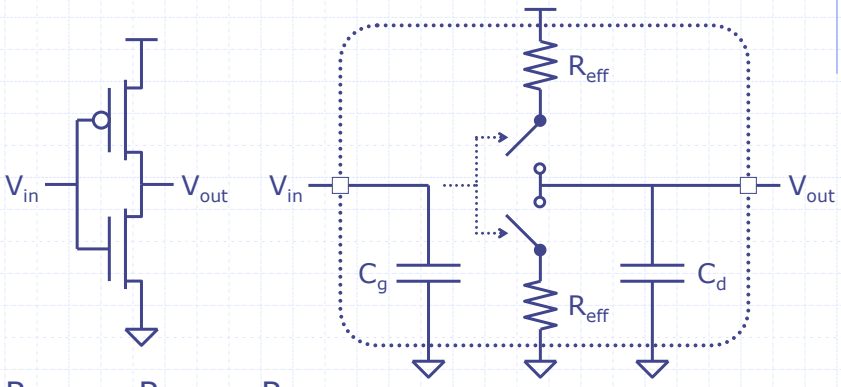
- Increase Width (W) ⇒ Increase current ⇒ Decrease R_{eff}
- Increase Length (L) ⇒ Decrease current ⇒ Increase R_{eff}
- C_{gate} proportional to (W x L) and C_{drain} proportional to W

The most basic CMOS gate is an inverter





RC model for an inverter

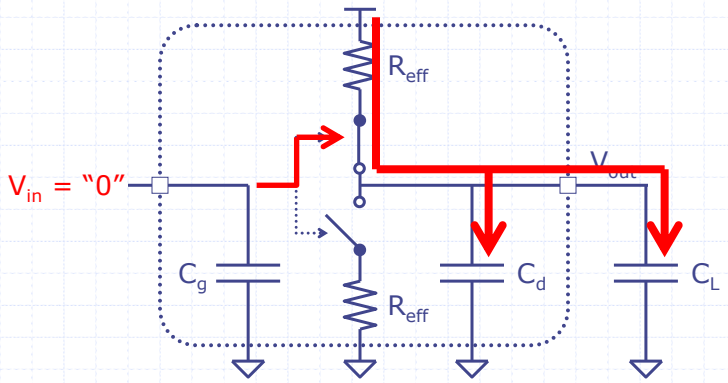


$R_{eff} = R_{eff,N} = R_{eff,P}$

$C_g = C_{g,N} + C_{g,P}$

$C_d = C_{d,N} + C_{d,P}$

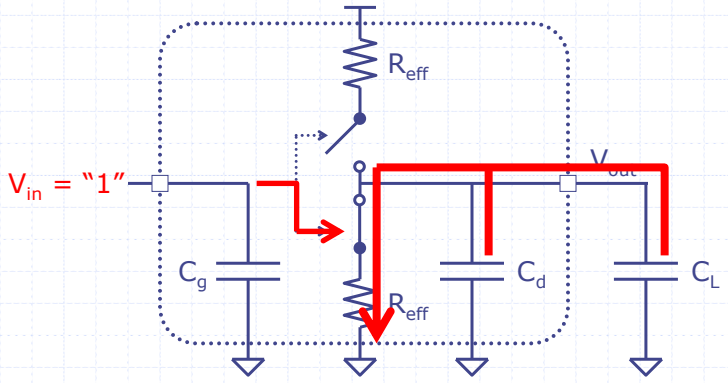
Charging time (0 → 1)



Charge RC Time Constant

$T_{PLH} = R_{eff} \times (C_d + C_L)$

Discharging time (1 → 0)



Discharge RC Time Constant

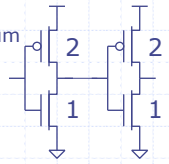
$T_{PHL} = R_{eff} \times (C_d + C_L)$

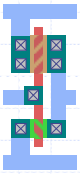
Larger gates are faster

decrease R_{eff} (but increase C_d!)

Process gen = 0.25µm
Supply voltage = 5V
Min width NMOS = 0.5µm

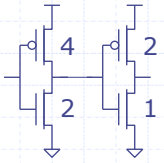
Param	Value	Units
$C_{d,N}$/µm	1.42	fF/µm
$C_{d,P}$/µm	2.40	fF/µm
$C_{g,N}$/µm	1.55	fF/µm
$C_{g,P}$/µm	1.48	fF/µm
$R_{eff,N}$ x µm	4.93	kΩ/µm
$R_{eff,P}$ x µm	10.83	kΩ/µm

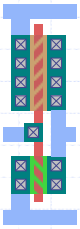




C_d = (0.5x1.42) + (1x2.40) = 3.11 fF
C_L = (0.5x1.55) + (1x1.48) = 2.26 fF
C_d+C_L = 5.37 fF
T_{PLH} = 2.2 x (10.83/1) x 5.37 = 128ps
T_{PHL} = 2.2 x (4.93/0.5) x 5.37 = 116ps

Double size of driver



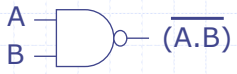


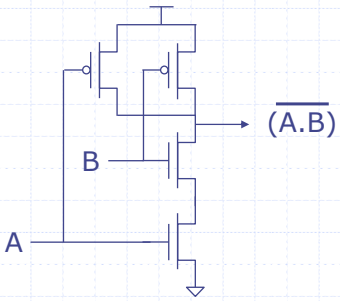
C_d = (1x1.42) + (2x2.40) = 3.66 fF
C_L = (0.5x1.55) + (1x1.48) = 2.26 fF
C_d+C_L = 5.92 fF
T_{PLH} = 2.2 x (10.83/2) x 5.92 = 70.5ps
T_{PHL} = 2.2 x (4.93/1) x 5.92 = 64.2ps

Ignores the fact that previous gate now must drive a bigger gate capacitance!

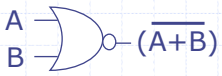
Bigger gates: NAND, NOR

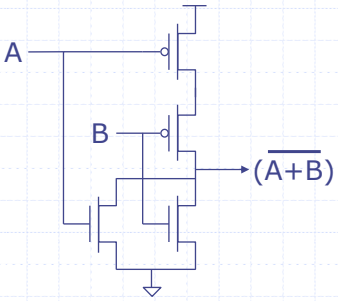
NAND Gate



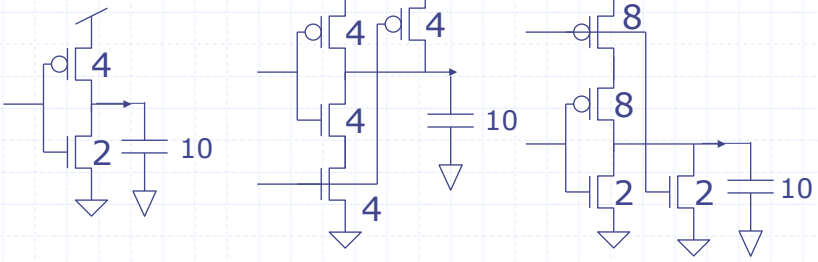


NOR Gate





Unit-less delay (d) of gates with equal drive strength (Reff)



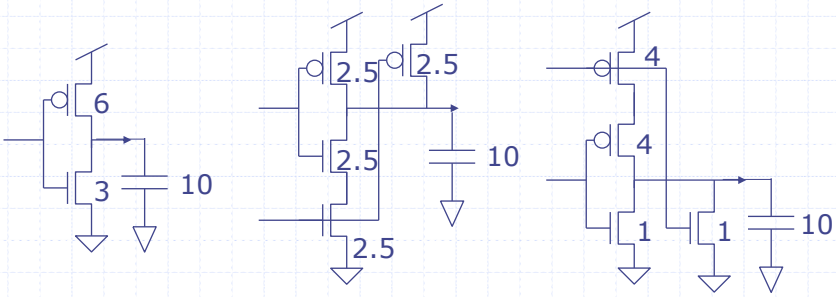
Inverter
delay = 2.67

Less parasitic drain capacitance (C_d) loading output

NAND
delay = 3.67

NOR
delay = 3.67

Unit-less delay (d) of gates with similar area



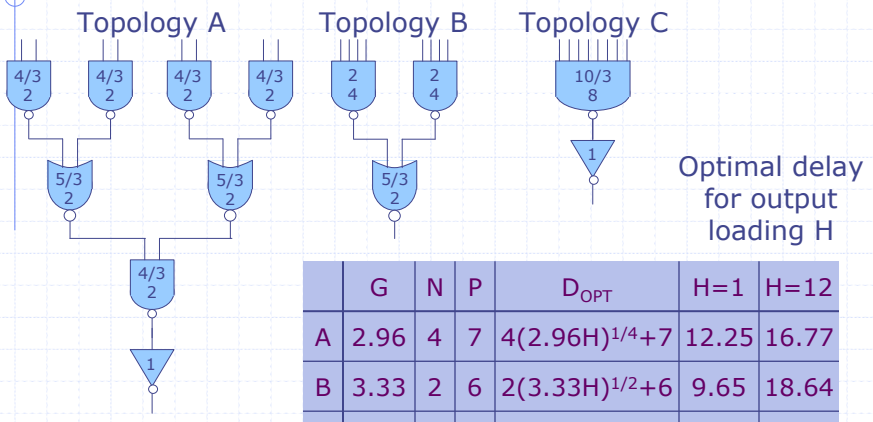
Inverter
delay = 2.11

NAND
delay = 4.67

NOR
delay = 5.33

PMOS worse than NMOS, series path is limiter

Optimal sizing and delays for example topologies

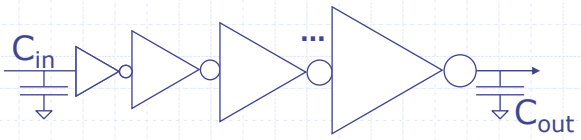


Optimal delay for output loading H

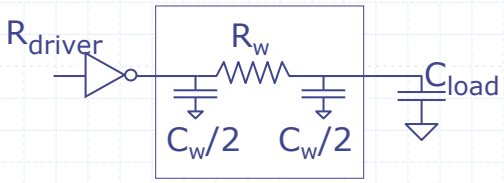
	G	N	P	D_{OPT}	H=1	H=12
A	2.96	4	7	$4(2.96H)^{1/4}+7$	12.25	16.77
B	3.33	2	6	$2(3.33H)^{1/2}+6$	9.65	18.64
C	3.33	2	9	$2(3.33H)^{1/2}+9$	12.65	21.64

[For more explanation of how these numbers were derived, see Logical Effort link in lab handout]

How many stages of inverters are required to drive a load?



A Lumped Π model of a wire



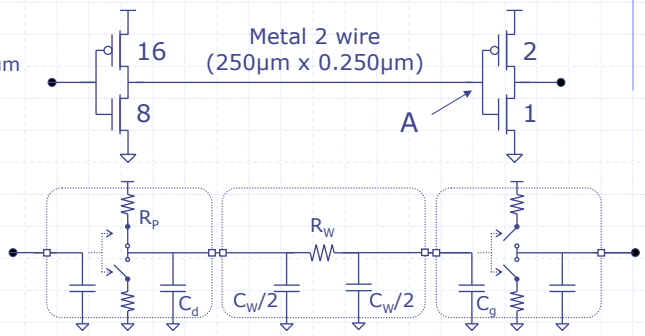
$$\text{Delay} \propto R_{driver} \times \frac{C_w}{2} + (R_{driver} + R_w) \times \left(\frac{C_w}{2} + C_{load} \right)$$

- R_w is lumped resistance of the wire
- C_w is lumped capacitance;
  - Partition half of C_w at each end

Estimate the rise time of node A

Process gen = 0.25µm
Supply voltage = 5V
Min width NMOS = 0.5µm

Param	Value	Units
$C_{d,N}$ / µm	1.42	fF/µm
$C_{d,P}$ / µm	2.40	fF/µm
$C_{g,N}$ / µm	1.55	fF/µm
$C_{g,P}$ / µm	1.48	fF/µm
$C_{A,M2}$ / µm²	0.016	fF/µm²
$C_{L,M2}$ / µm	0.084	fF/µm
$R_{eff,N}$ x µm	4.93	kΩ/µm
$R_{eff,P}$ x µm	10.83	kΩ/µm
R_{M2} / sq	0.07	Ω/sq

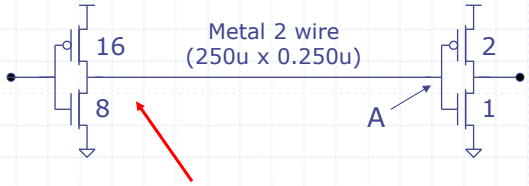


C_g = (0.5 x 1.55) + (1 x 1.48) = 2.26 fF
C_d = (4 x 1.42) + (8 x 2.40) = 24.88 fF
R_p = 10.83/8 = 1.35 kΩ
R_w = (250 / 0.25) x 0.07 = 70 Ω
C_w = ((250 x 0.25) x 0.0016)+(250 x 0.084) = 21.14 fF
T_{PLH} = 2.2 x (1350 x (21.14/2 + 24.88)
+ (1350 + 70) x (21.14/2 + 2.26)) = 66ps

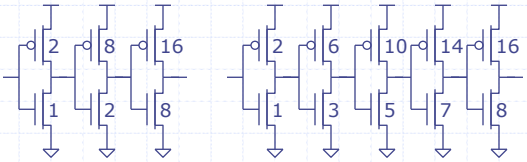
Adding buffers

Process gen = 0.25µm
Supply voltage = 5V
Min width NMOS = 0.5µm

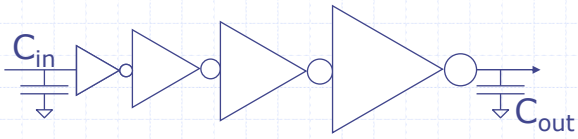
Param	Value	Units
$C_{d,N}$ / µm	1.42	fF/µm
$C_{d,P}$ / µm	2.40	fF/µm
$C_{g,N}$ / µm	1.55	fF/µm
$C_{g,P}$ / µm	1.48	fF/µm
$C_{A,M2}$ / µm²	0.016	fF/µm²
$C_{L,M2}$ / µm	0.084	fF/µm
$R_{eff,N}$ x µm	4.93	kΩ/µm
$R_{eff,P}$ x µm	10.83	kΩ/µm
R_{M2} / sq	0.07	Ω/sq



Should we have a few big stages or many small stages?



A good rule-of-thumb is to target a stage effort around four



Minimum delay when:

- Stage effort = logical effort x electrical effort ≈ 3.4-3.8
- Some derivations use e = 2.718.. – this ignores parasitics
- Broad optimum, stage efforts of 2.4-6.0 within 15-20% of minimum

Fan-out-of-four (FO4) is convenient design size *(~5τ)*

FO4 delay: Delay of inverter driving four copies of itself