
Computer Science 141

Computing Hardware

Fall 2009

Harvard University

Instructor: Prof. David Brooks

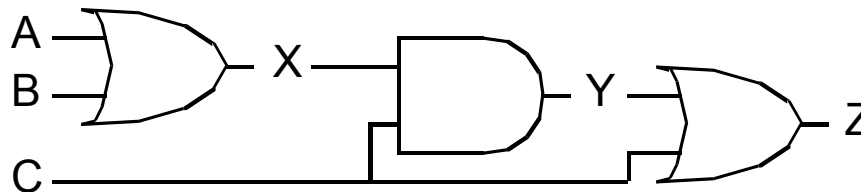
dbrooks@eecs.harvard.edu

Today's Lecture

- Time Response/Glitches
- Digital Design Components

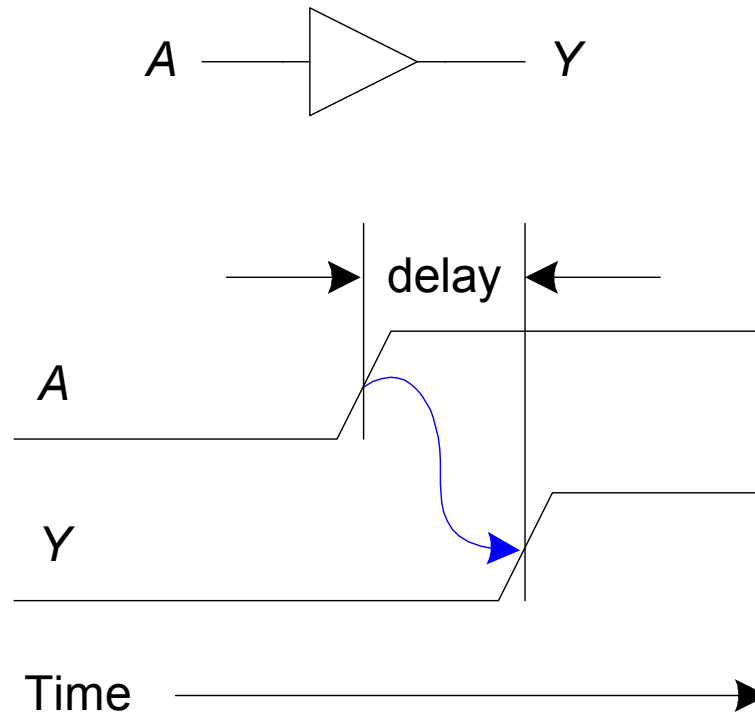
Time Response in Combinational Circuits

- Circuit analysis so far concerned only with steady state behavior
 - apply input pattern, observe output pattern (given enough time)
- Dynamic behavior important too
 - propagation delay – varies depending upon
 - path of change through circuit
 - direction of change within gate



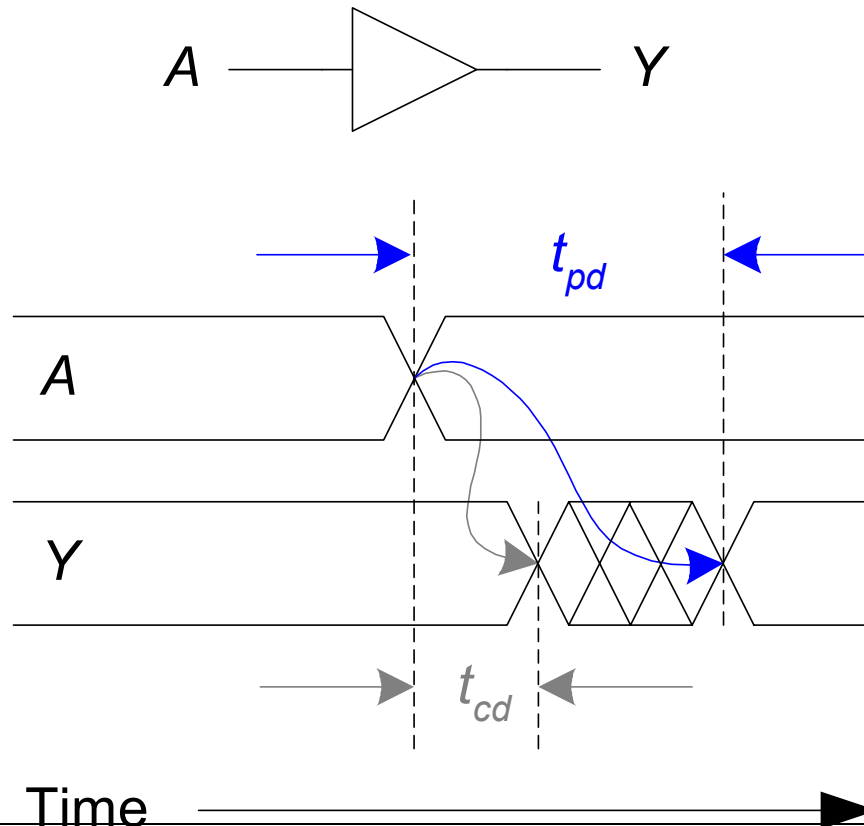
Timing

- Delay between input change and output changing
- How to build fast circuits?



Propagation & Contamination Delay

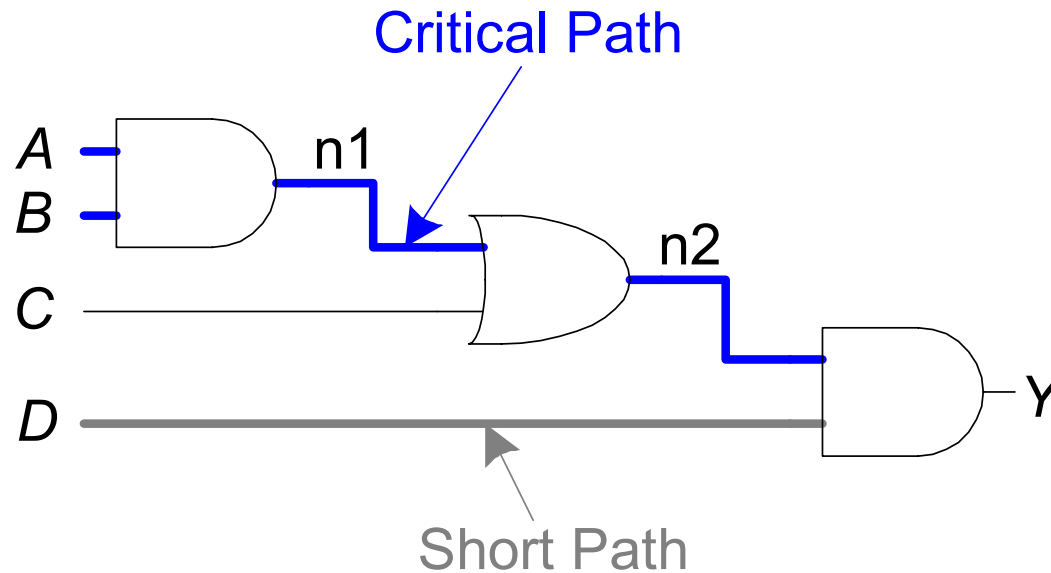
- Propagation delay: $t_{pd} = \max$ delay from input to output
- Contamination delay: $t_{cd} = \min$ delay from input to output



Propagation & Contamination Delay

- Delay is caused by
 - Capacitance and resistance in a circuit
 - Speed of light limitation
- Reasons why t_{pd} and t_{cd} may be different:
 - Different rising and falling delays
 - Multiple inputs and outputs, some of which are faster than others
 - Circuits slow down when hot and speed up when cold

Critical (Long) and Short Paths



Critical (Long) Path: $t_{pd} = 2t_{pd_AND} + t_{pd_OR}$

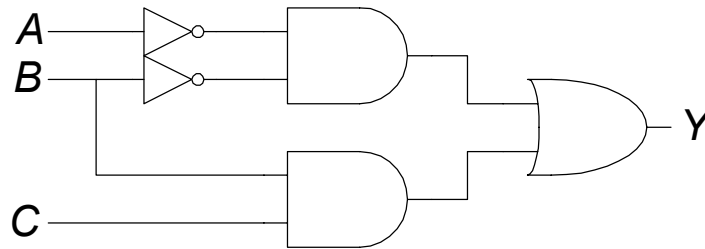
Short Path: $t_{cd} = t_{cd_AND}$

Glitches

- Glitch: when a single input change causes multiple output changes
- Glitches don't cause problems because of synchronous design conventions (which we'll talk about in a bit)
- But it's important to recognize a glitch when you see one in timing diagrams

Glitch Example

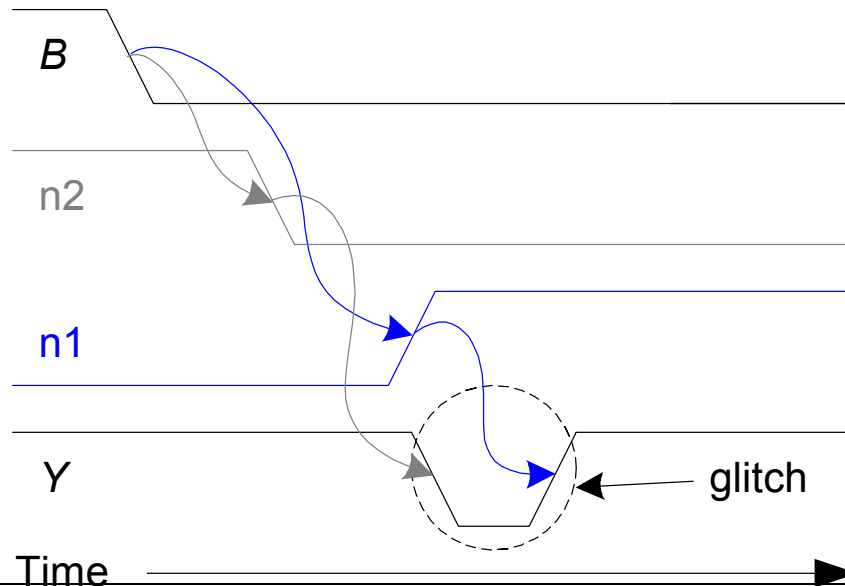
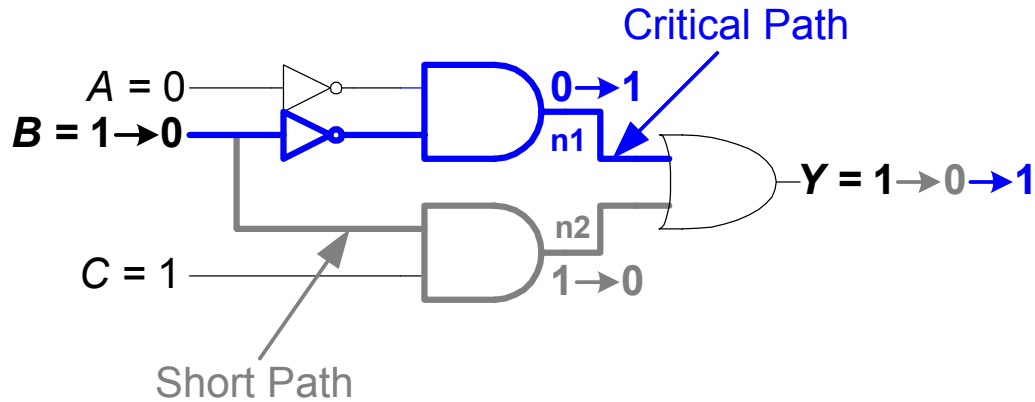
- What happens when $A = 0$, $C = 1$, B falls?



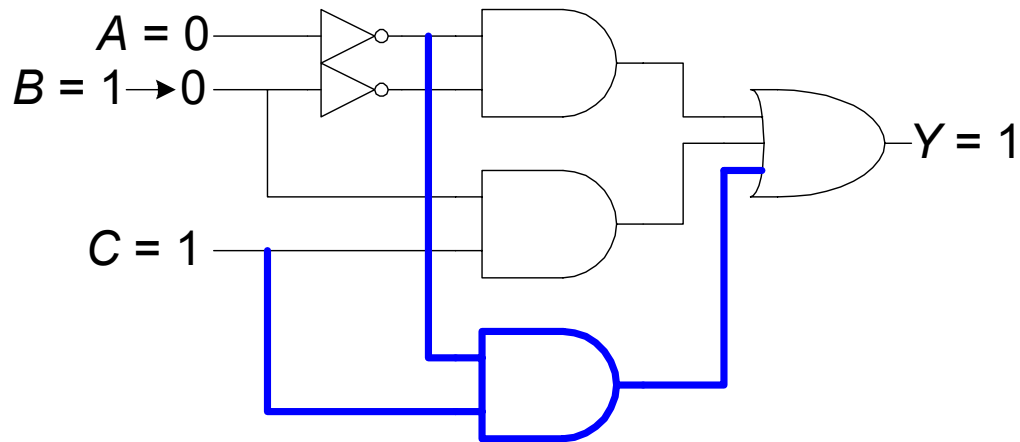
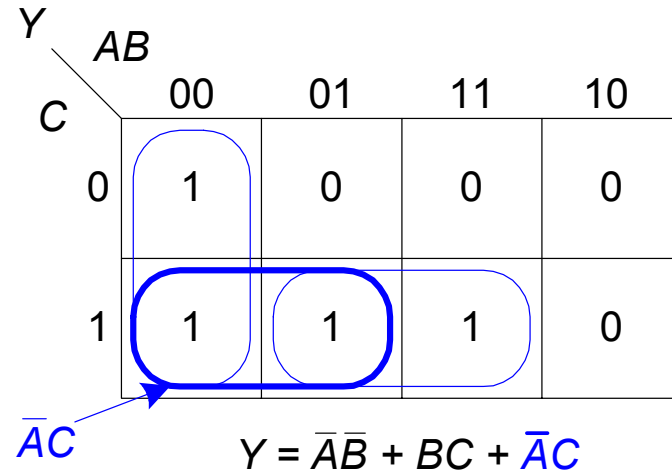
Y		AB			
		00	01	11	10
C	0	1	0	0	0
	1	1	1	1	0

$$Y = \bar{A}\bar{B} + BC$$

Glitch Example (cont.)



Fixing the Glitch



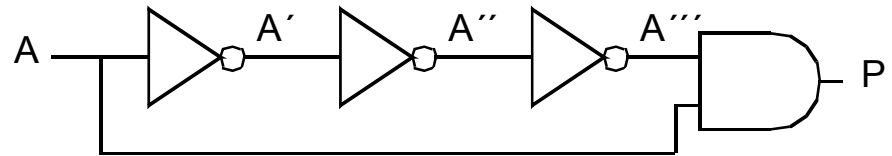
Why Understand Glitches?

- Glitches don't cause problems because of synchronous design conventions (Chapter 3)
- But it's important to recognize a glitch when you see one in simulations or on an oscilloscope
- Can't get rid of all glitches – simultaneous transitions on multiple inputs can also cause glitches

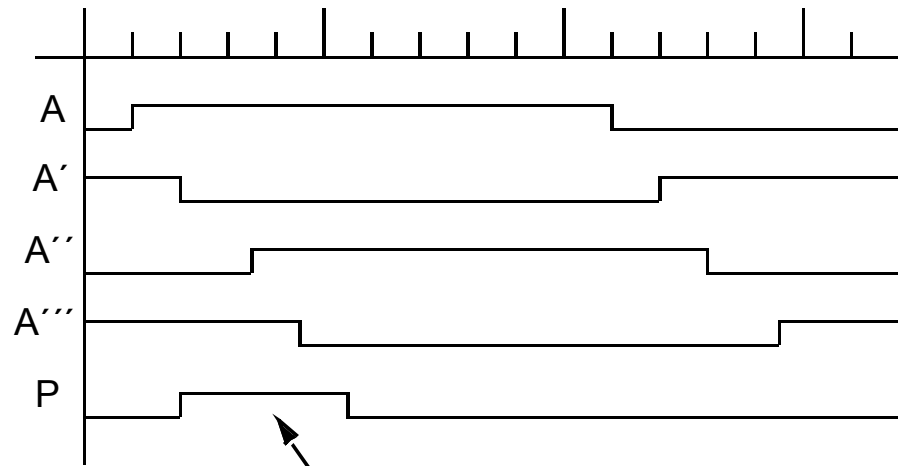
Useful circuits employing delay

1. Pulse shaping circuit

- statically boring, dynamically useful
- Takes advantage of asymmetry in delay paths
- length of pulse depends on number of inverters
- circuit not as interesting if even number of inverters



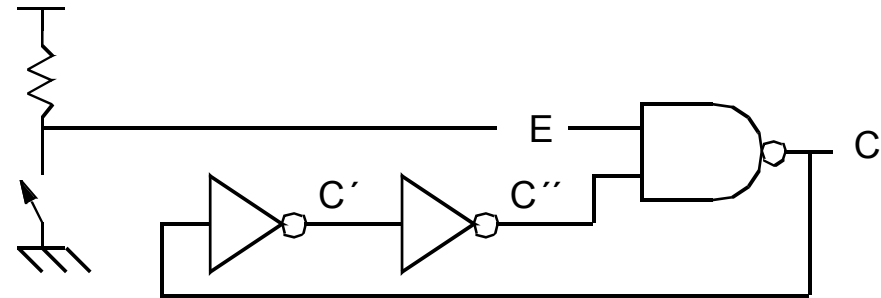
$$P = A \cdot A''' = A \cdot A' = 0$$



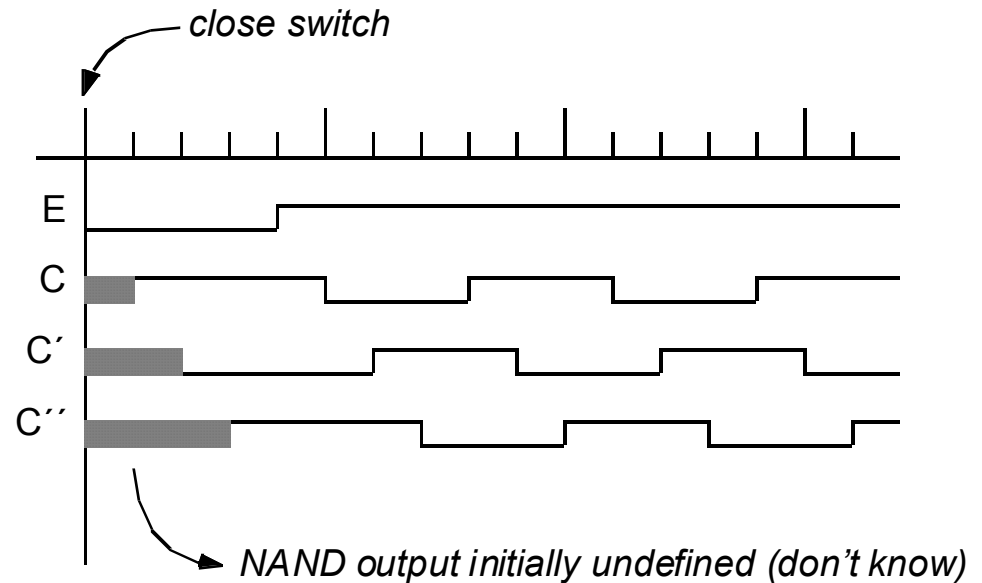
Useful circuits employing delay

2. Oscillator

- odd number of inverters in a loop produces a continually-running pulse generator



- add ENABLE pin to reset oscillator

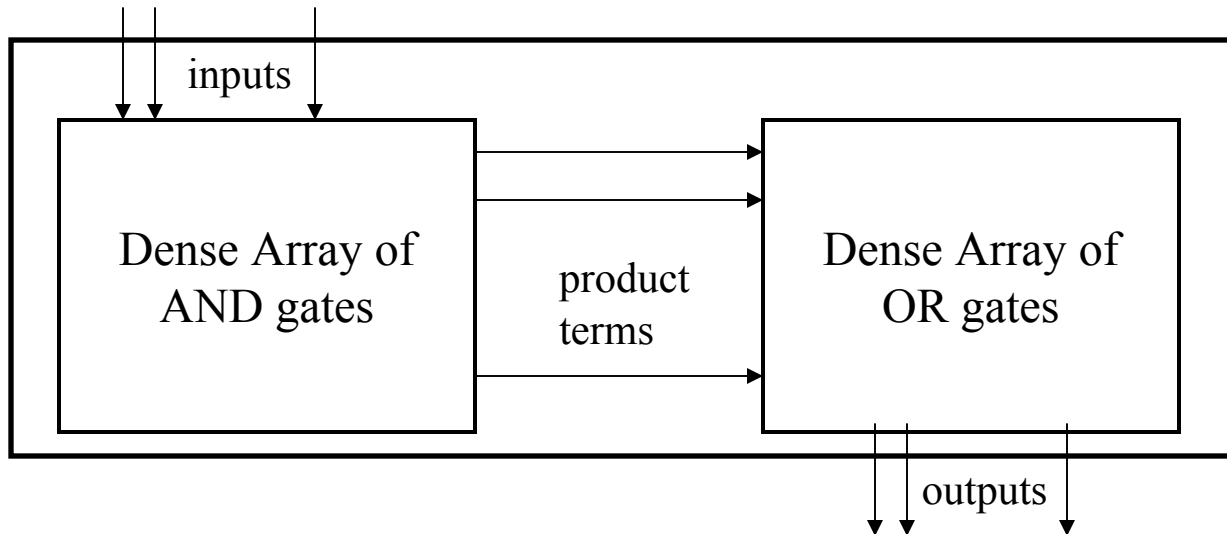


Key Digital Design Components

- Programmable Logic and ROMs
- Steering Logic (Multiplexors/Demultiplexors) and Decoders
 - How to implement logic with multiplexers and decoders
- Bus Design
 - Describe gates with outputs other than 0 and 1 (tri-state and OC)
- Arithmetic Circuits
 - Adders/Subtractors
 - Multipliers

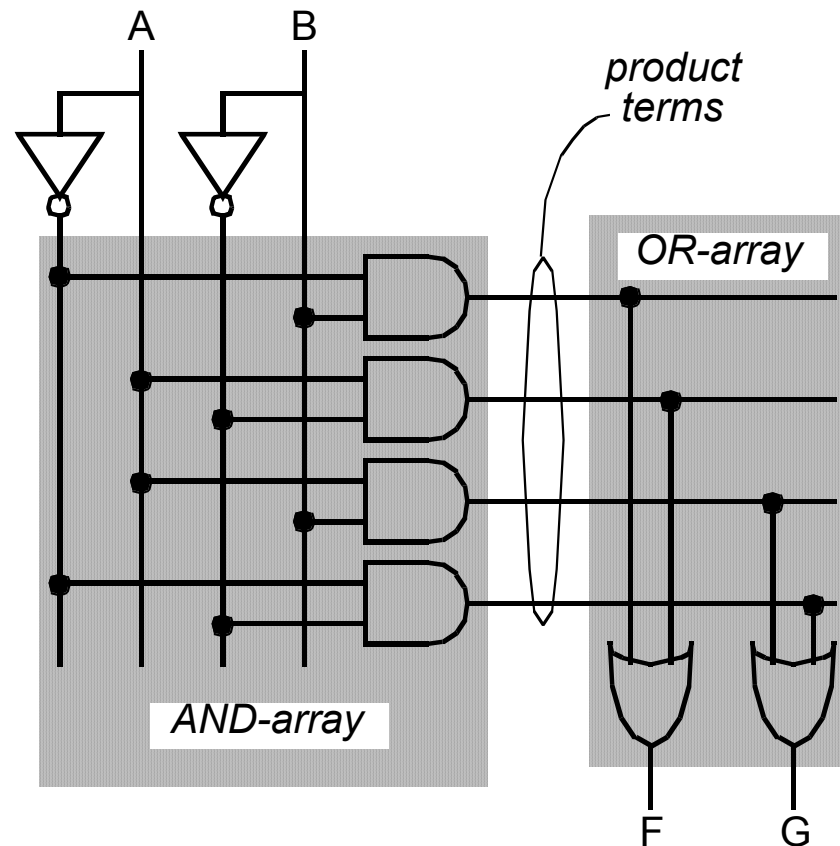
Programmable and Steering Logic

- Overview
 - Programmable logic devices (PLAs, PALs, ROMs)
 - customizable, general-purpose building blocks
 - complex functions in very little space

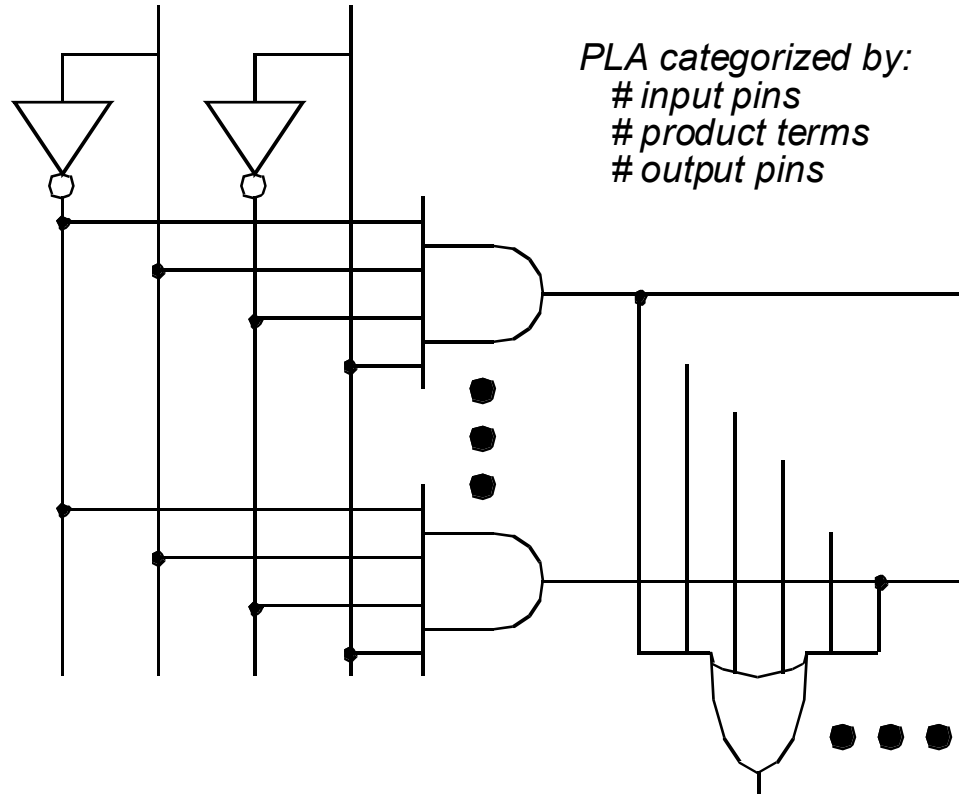


Programmable Logic Arrays

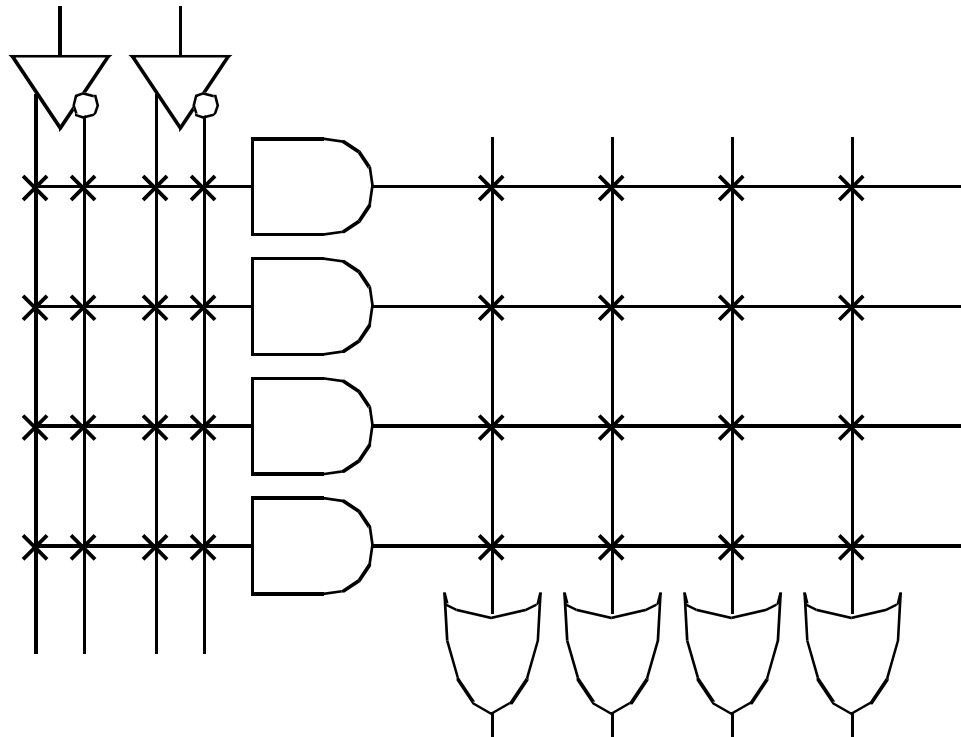
- motivated from SOPs 2-level canonical form



PLA before programming



shorthand notation for PLA



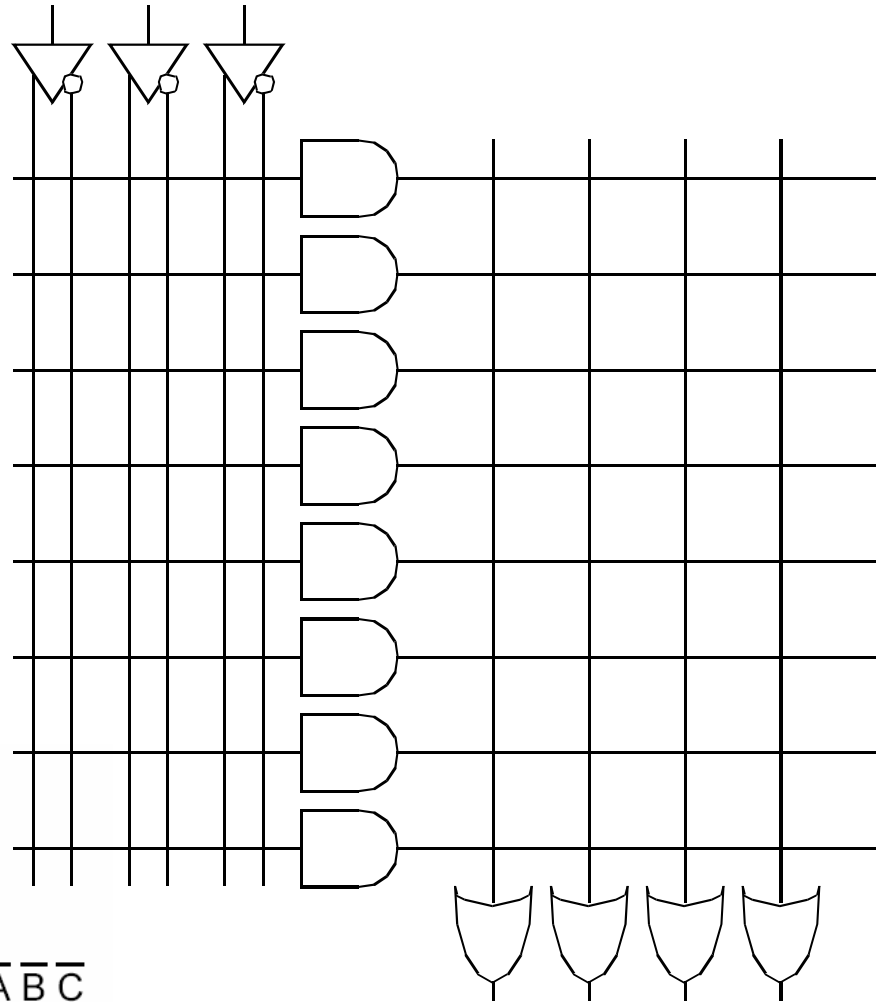
PLA design example

$$F0 = A \cdot B \cdot C$$

$$F1 = A + B + C$$

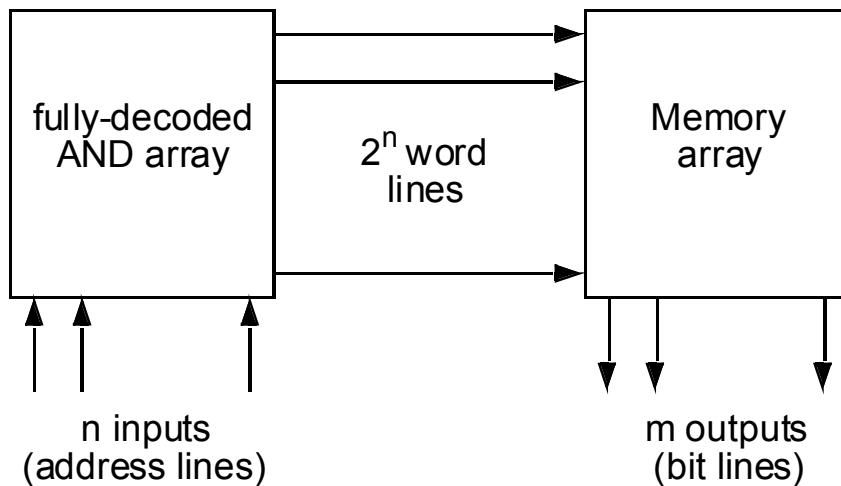
$$F2 = (A + B + C)' = \bar{A} \cdot \bar{B} \cdot \bar{C}$$

$$F3 = (A \oplus B \oplus C)' = \bar{A} B C + A \bar{B} C + A B \bar{C} + \bar{A} \bar{B} \bar{C}$$

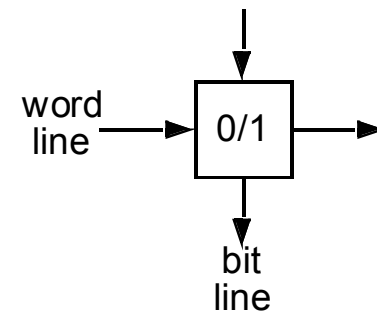


Read-Only Memory (ROM)

- conceptually, PLA with fully-decoded AND array
 - (i.e. every possible minterm available)

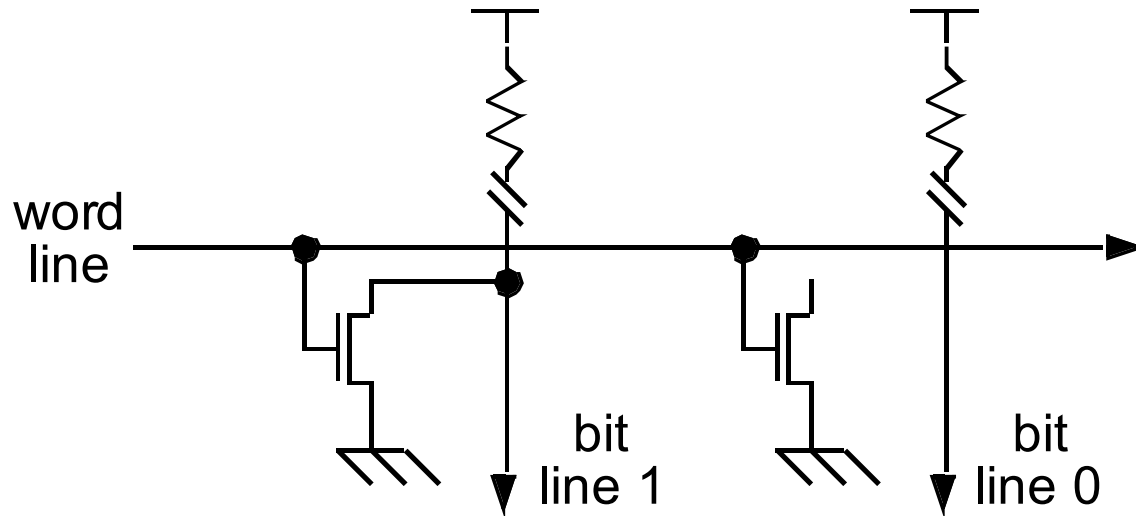


- OR-array replaced by memory array
 - each cell in memory array stores TT value for that minterm



Read-Only Memory (ROM)

- physically implemented with transistor to GND, bit line pulled up



8 word x 4 bit ROM design example

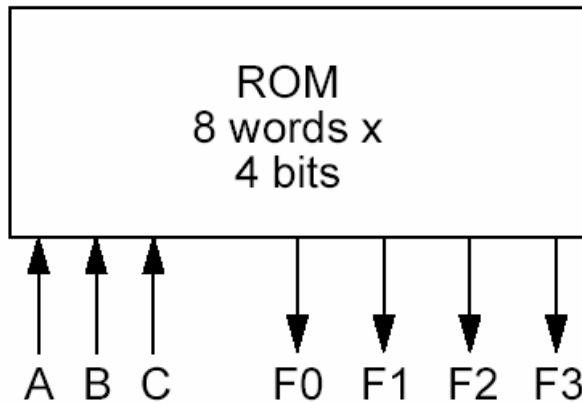
ROM design example

$$F0 = A \cdot B \cdot C$$

$$F1 = A + B + C$$

$$F2 = (A + B + C)' = \bar{A} \cdot \bar{B} \cdot \bar{C}$$

$$F3 = (A \oplus B \oplus C)' = \bar{A} B C + A \bar{B} C + A B \bar{C} + \bar{A} \bar{B} \bar{C}$$



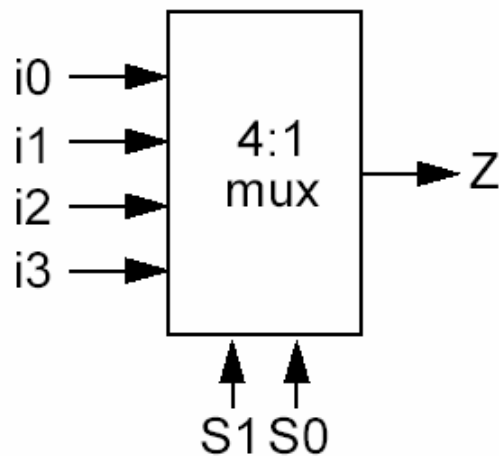
A	B	C	F0	F1	F2	F3
0	0	0	0	0	1	1
0	0	1	0	1	0	0
0	1	0	0	1	0	0
0	1	1	0	1	0	1
1	0	0	0	1	0	0
1	0	1	0	1	0	1
1	1	0	0	1	0	1
1	1	1	1	1	0	0

ROM
address

word
contents

Steering Logic

- Multiplexer (selector)
 - drive one of n inputs onto single output

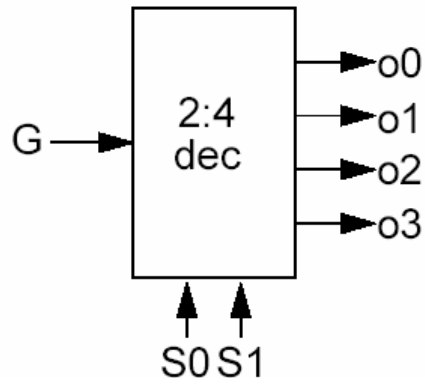


S_1	S_0	Z
0	0	i_0
0	1	i_1
1	0	i_2
1	1	i_3

- 2^n inputs, n selector lines, 1 output $\Rightarrow 2^n:1$ multiplexer

decoder (demultiplexer)

- drive single input onto one of n outputs



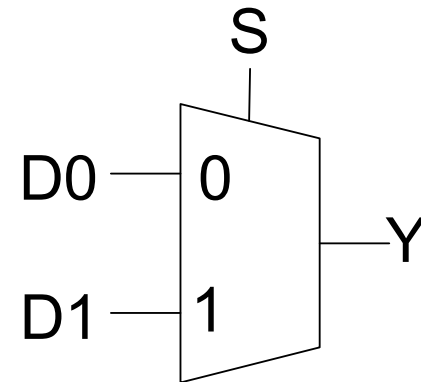
S1	S0	o0	o1	o2	o3
0	0	G	0	0	0
0	1	0	G	0	0
1	0	0	0	G	0
1	1	0	0	0	G

- need default value for unasserted outputs
- 1 input, n decoder lines, 2^n outputs \Rightarrow $n:2^n$ decoder
- multiplexer/decoder can be implemented with transmission gates
 \Rightarrow cheaper in terms of transistor counts for large structures

Mux implementation

- 2:1 multiplexer chooses between two inputs

S	D1	D0	Y
0	X	0	
0	X	1	
1	0	X	
1	1	X	



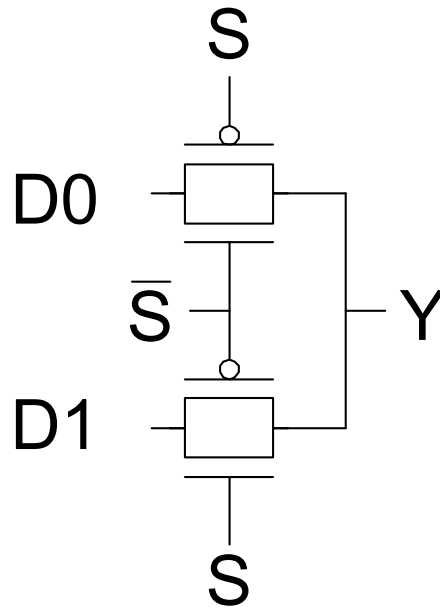
Y=

Mux Implementation

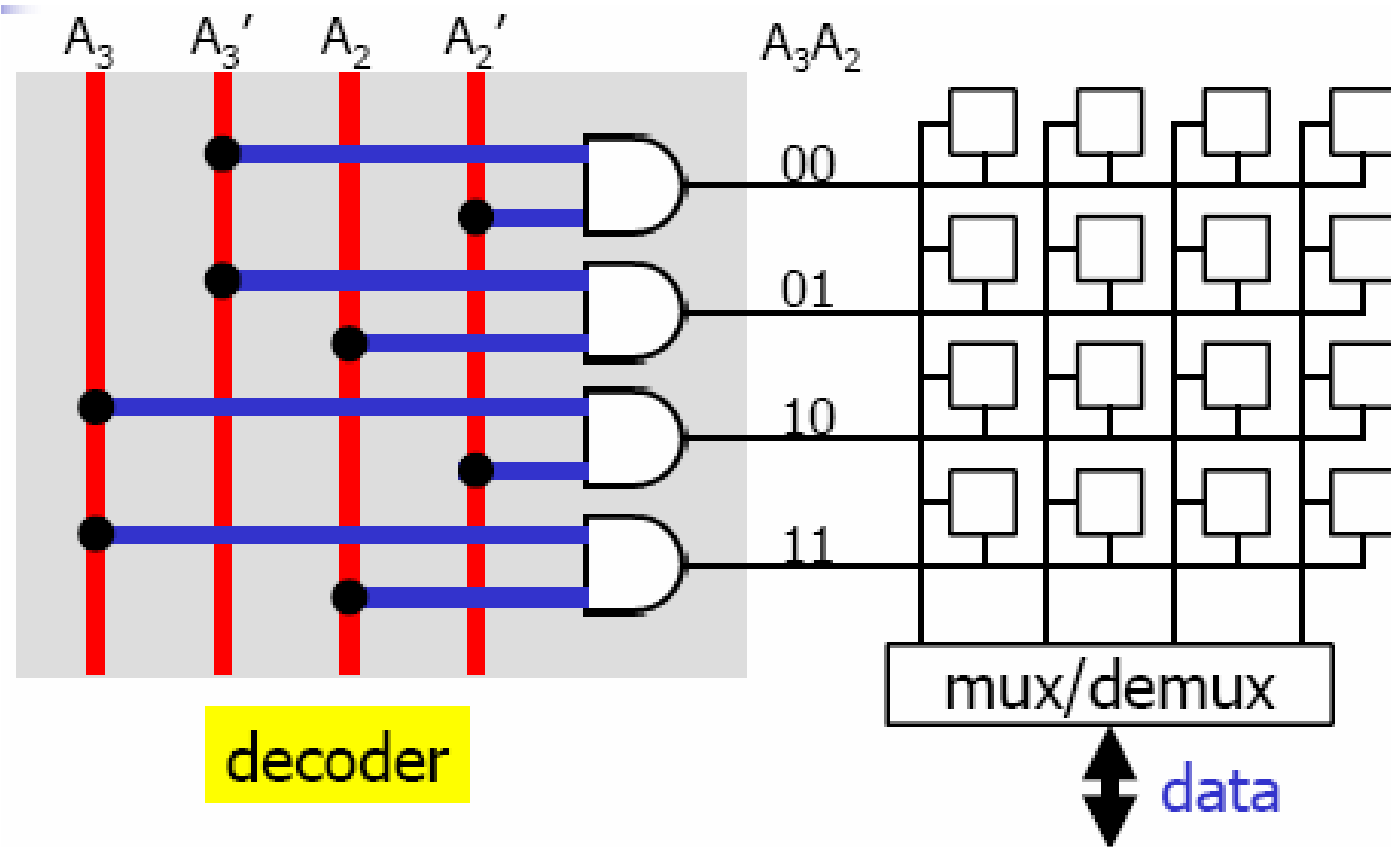
$$Y = SD_1 + \bar{S}D_0 \text{ (too many transistors)}$$

Transmission Gate Mux

- Nonrestoring mux uses two transmission gates
 - Only 4 transistors



Building a fast decoder

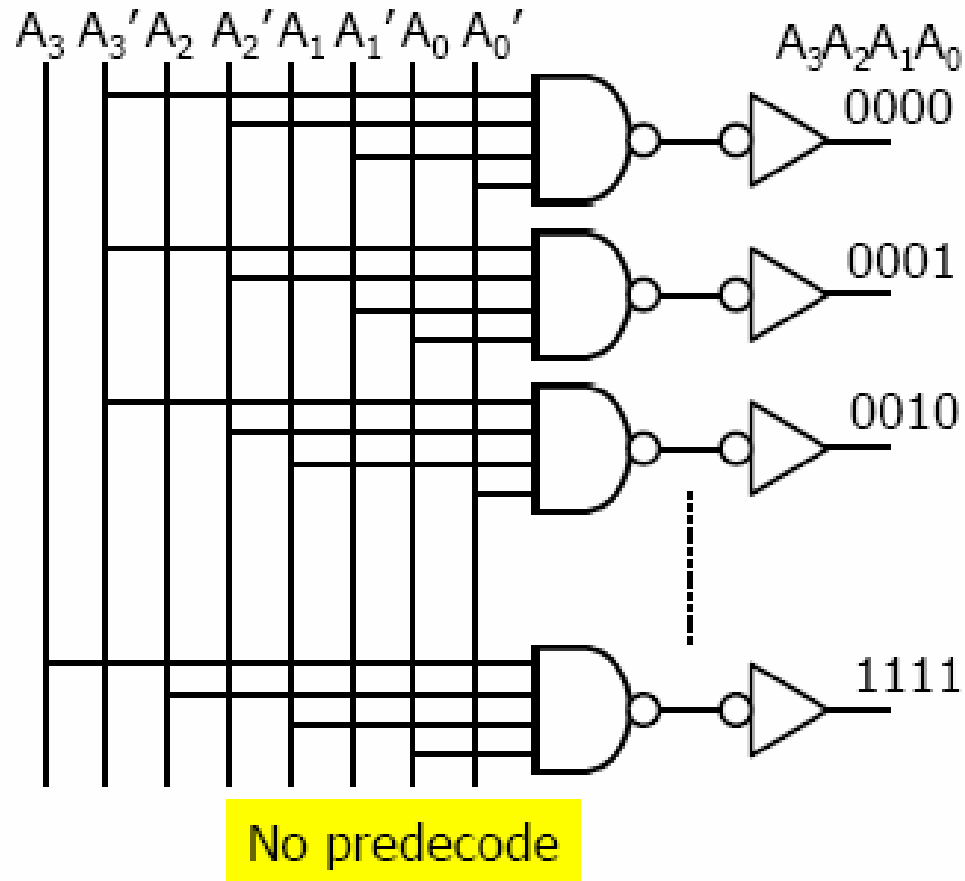


- For $n \rightarrow 2^n$ decoder, need 2^n n -input AND's
- Must be regular layout

Example from Stanford EE271 notes

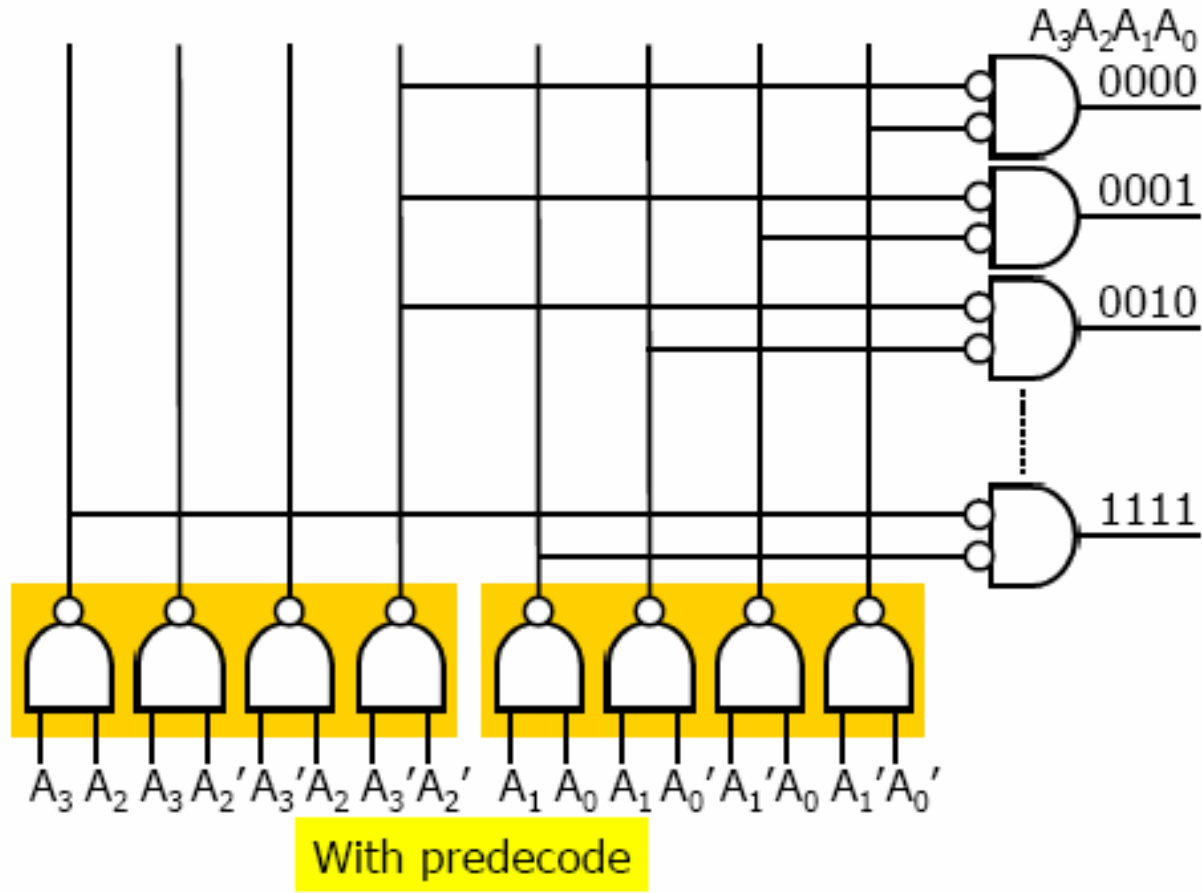
CMOS Decoder

- Large fanin AND gates are slow



Pre-decoding bits

- Use two-level decoder



More predecoding

- Predecode binary address into octal addresses
 - For example, 9-bit address ($A_8, A_7, A_6, \dots, A_0$) can be decoded in two-level implementation
 - No Predecode
 - Requires ____, ____-input AND gates
 - Predecode
 - Predecode $A_8 A_7 A_6$, $A_5 A_4 A_3$, and $A_2 A_1 A_0$ using ____, ____-input NANDs
 - Followed by ____ 3-input NOR's

Next Lecture

- More building blocks
 - Muxes/Decoders
 - Tri-state devices and buses
 - Arithmetic circuits
 - Adders