Computer Organization and Design

Transistors & Logic - II

Henry Fuchs

Slides adapted from Montek Singh, who adapted them
from Leonard McMillan and from Gary Bishop
Back to McMillan & Chris Terman, MIT 6.004 1999

Thursday, March 19, 2015

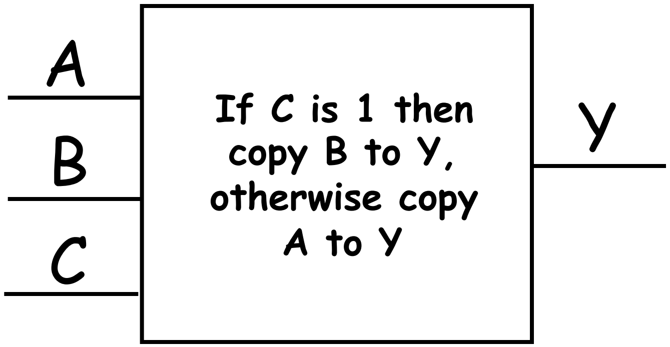
Lecture 10

Today's Topics

- Synthesis using standard gates
  - Truth tables
  - Universal gates: NAND and NOR
  - Gates with more than 2 inputs
  - Sum-of-Products
  - DeMorgan's Law

Now We're Ready to Design Stuff!

- We need to start somewhere
  - usually it's the functional specification



If you are like most engineers you'd rather see a table, or formula than parse a logic puzzle. The fact is, **any combinational function can be expressed as a table.**

These "**truth tables**" are a concise description of the combinational system's function. Conversely, any computation performed by a combinational system can expressed as a truth table.

Truth Table

C	B	A	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

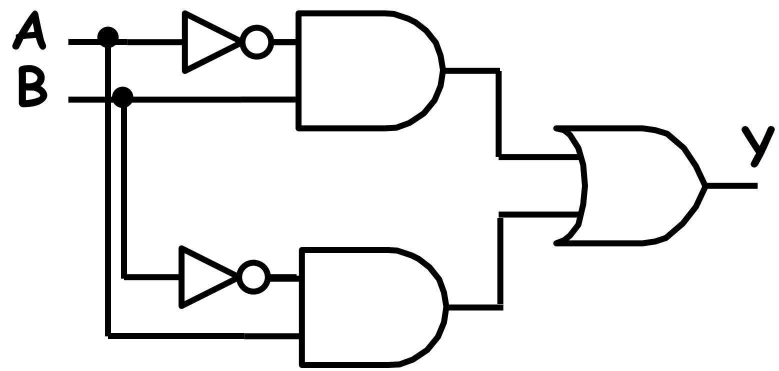
We Can Make Most Gates Out of Others

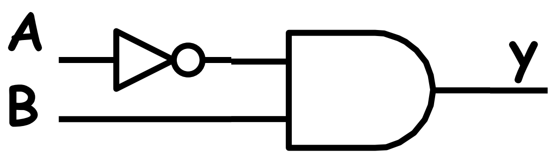
B>A

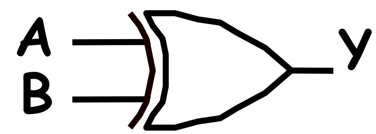
AB	Y
00	0
01	1
10	0
11	0

XOR

AB	Y
00	0
01	1
10	1
11	0



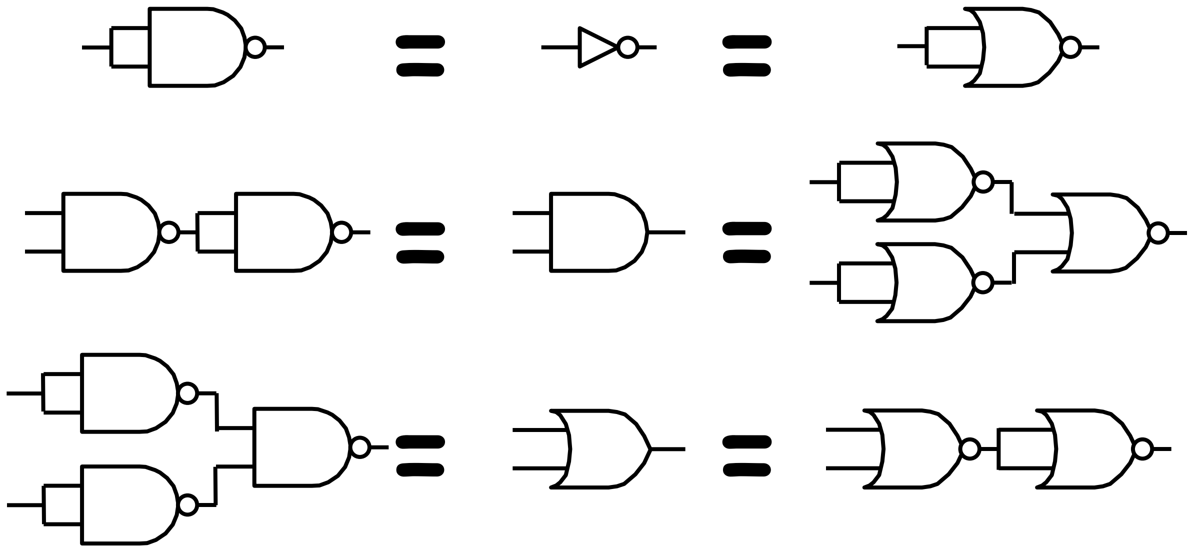




* How many different gates do we really need?

One Will Do!

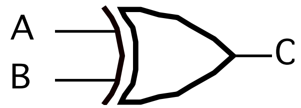
- NANDs and NORs are universal
  - one can make *any* circuit out of just NANDs, or just NORs!



- Ah! But what if we want more than 2-inputs?

Gate Trees

Suppose we have some 2-input XOR gates:
(same idea holds for AND and OR gates)

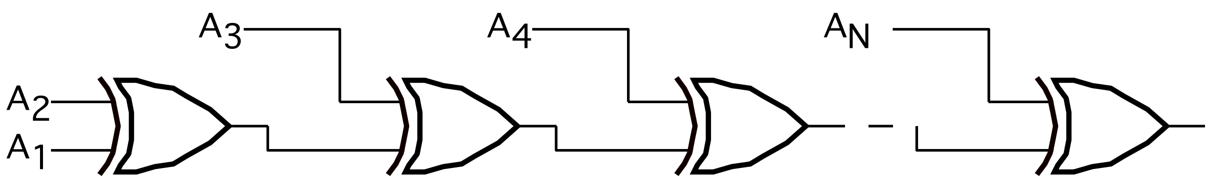


$t_{pd} = 1$
(latency)

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

output = 1
iff number of 1s
in input is ODD
("ODD PARITY")

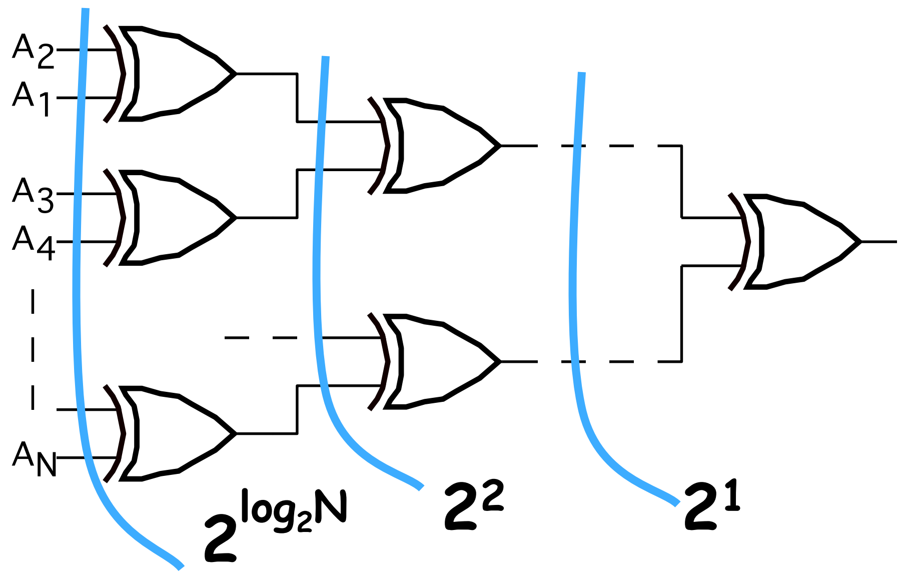
And we want an N-input XOR:



t_{pd} (latency)= O($\underline{N}$) -- WORST CASE.

Can we compute N-input XOR faster?

Gate Trees



N-input TREE has O(log N) levels...

Signal propagation takes O(log N) gate delays.

Design Approach: Sum-of-Products

Three steps:

1. Write functional spec as a truth table
2. Write down a Boolean expression for every ‘1’ in the output

$$Y = \overline{C}\,\overline{B}A + \overline{C}BA + CB\overline{A} + CBA$$

3. Wire up the gates!

Truth Table

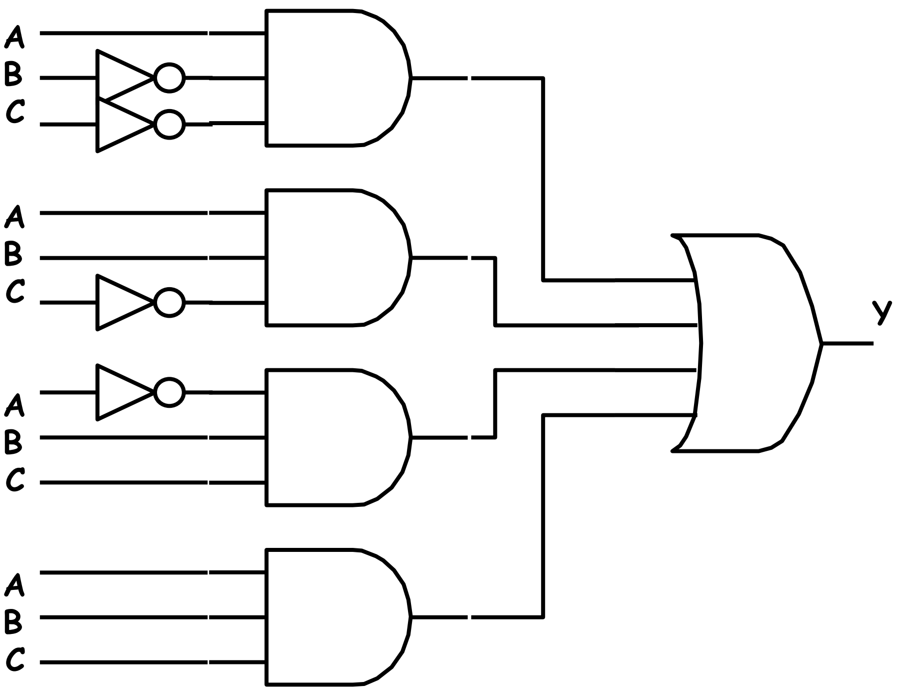
C	B	A	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

* This approach will always give us logic expressions in a particular form:
  - SUM-OF-PRODUCTS (“SOP”)
    - “SUM” actually means OR
    - “PRODUCT” actually means AND

Straightforward Synthesis

- We can implement SUM-OF-PRODUCTS…
  - …with just three levels of logic:
    - INVERTERS/AND/OR

$$Y = \overline{C}\,\overline{B}A + \overline{C}BA + CB\overline{A} + CBA$$



Notations

- Symbols and Boolean operators:

$$x \cdot y, \;\; xy, \;\; x \wedge y, \;\; \mathrm{AND}(x,y), \;\; x \;\; \mathrm{AND} \;\; y$$

$$x + y, \;\; x \vee y, \;\; \mathrm{OR}(x,y), \;\; x \;\; \mathrm{OR} \;\; y$$

$$\overline{x}, \;\; x', \;\; \neg x, \;\; \mathrm{NOT}(x), \;\; \mathrm{INV}(x)$$

$$\overline{x \cdot y}, \;\; \overline{x \wedge y}, \;\; \overline{xy}, \;\; \mathrm{NAND}(x,y), \;\; x \;\; \mathrm{NAND} \;\; y$$

$$\overline{x + y}, \;\; \overline{x \vee y}, \;\; \mathrm{NOR}(x,y), \;\; x \;\; \mathrm{NOR} \;\; y$$

$$x \oplus y, \;\; \mathrm{XOR}(x,y), \;\; x \;\; \mathrm{XOR} \;\; y$$

$$x \,\overline{\oplus}\, y, \;\; \overline{x \oplus y}, \;\; \mathrm{XNOR}(x,y), \;\; x \;\; \mathrm{XNOR} \;\; y$$

DeMorgan's Laws

- Change ANDs into ORs and vice-versa

$$\overline{x + y} = \overline{x} \cdot \overline{y}$$

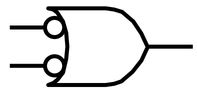
$$\overline{x \cdot y} = \overline{x} + \overline{y}$$

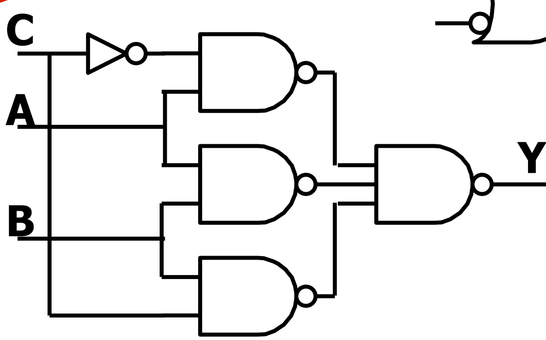
Useful Gate Structures

$\overline{C}A + AB + BC$

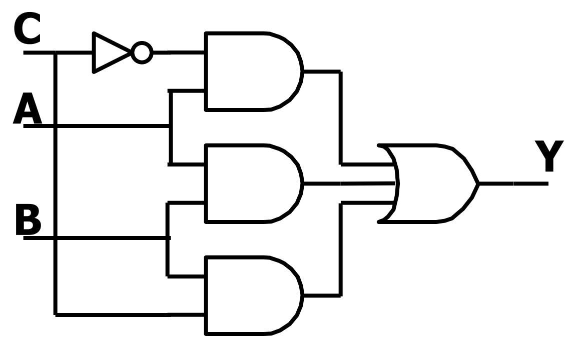
- NAND-NAND

$\overline{AB} = \overline{A} + \overline{B}$





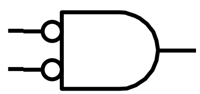
"Pushing Bubbles"



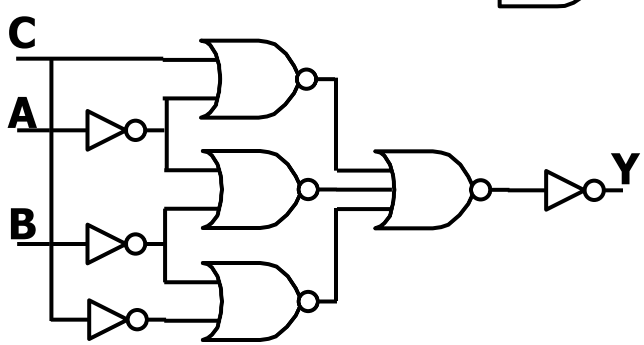
$\overline{xyz} = \bar{x} + \bar{y} + \bar{z}$

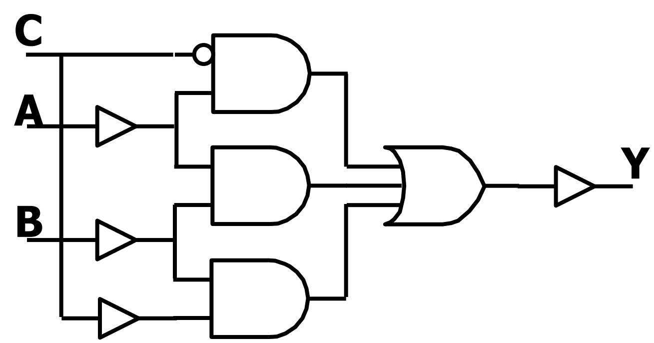
DeMorgan's Laws

$\overline{A}\,\overline{B} = \overline{A + B}$



- NOR-NOR

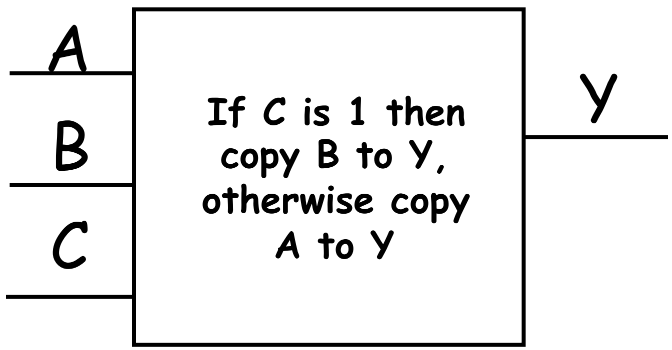




$\overline{x + y} = \bar{x}\bar{y}$

An Interesting 3-Input Gate: Multiplexer

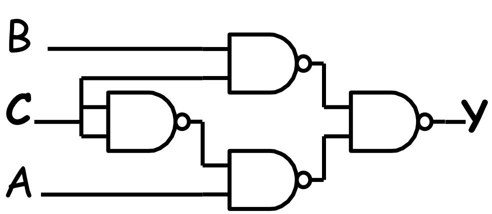
* Based on C, select the A or B input to be copied to the output Y.



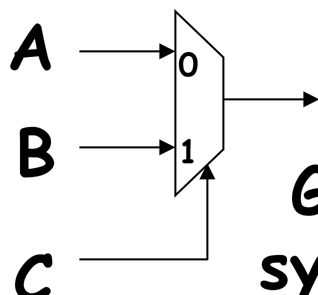
2-input Multiplexer

Truth Table

C	B	A	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



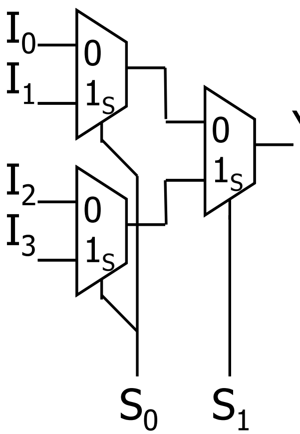
"schematic" diagram

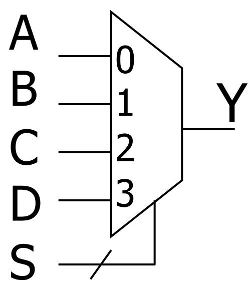


Gate symbol

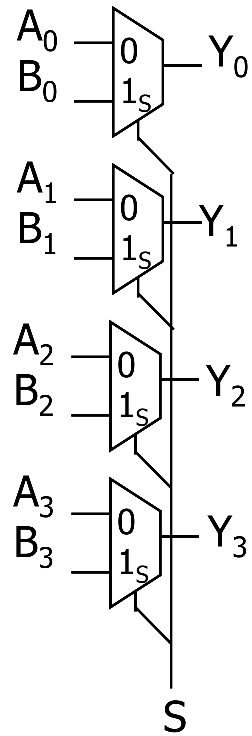
Multiplexer (MUX) Shortcuts

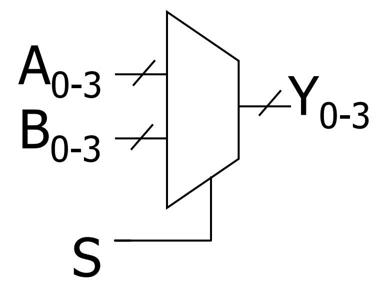
A 4-input Mux
(implemented as a tree)





A 4-bit wide 2-input Mux





Next Class

- Arithmetic circuits