All the Gates

There are only 16 possible 2-input gates... Let's examine all of them. Some we already know, others are just silly.

INPUT AB	ZERO	AND	A > B	A	B > A	B	XOR	OR	NOR	XNOR	NOT 'B'	A <= B	NOT 'A'	B <= A	NAND	ONE
00	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	1	1	1	1	1	1	1	1
01	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
11	0	~~1~~	0	~~1~~	0	~~1~~	0	~~1~~	0	~~1~~	0	~~1~~	0	~~1~~	0	~~1~~

How many of these gates can be implemented using a single CMOS gate?

N-FETs can only pull the ouput to "0", and only if one or more of their inputs is a "1".

P-FETs can only pull the ouput to "1", and only if one or more of their inputs is a "0".

Do we really need all of these gates?

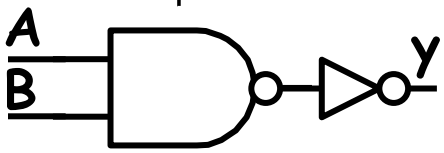
Nope! Once we realize that we can describe all of them using just NOT, followed by AND, followed by OR.

Composing gates to build others

Let's start with a couple of basics, AND and OR. Each can be constructed using a pair of CMOS gates, AND is just NAND with an inverter, and OR is just NOR with an inverted output.

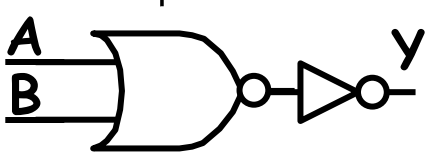
AND

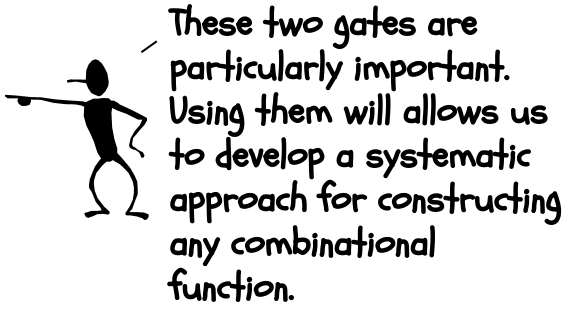
AB	Y
00	0
01	0
10	0
11	1



OR

AB	Y
00	0
01	1
10	1
11	1





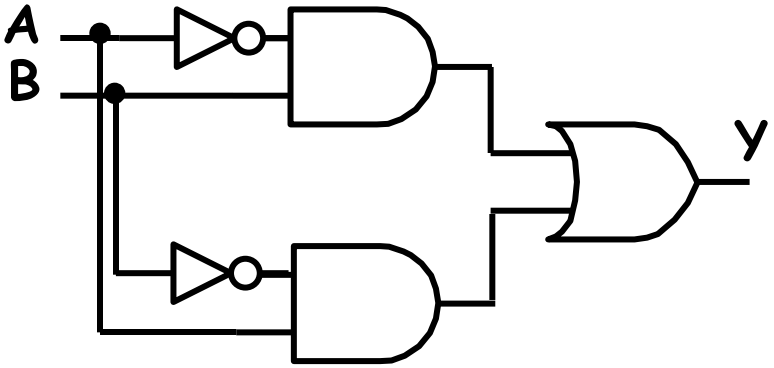
Composing Arbitrary Gates

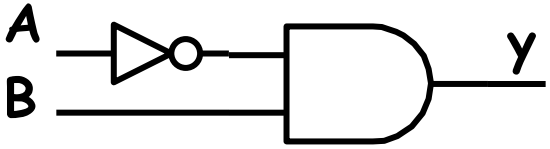
B>A

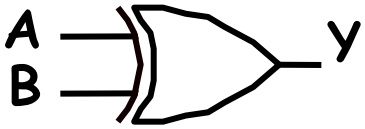
AB	Y
00	0
01	1
10	0
11	0

XOR

AB	Y
00	0
01	1
10	1
11	0







How many different gates do we really need?

We can always do it with 3 different types of gates (AND, OR, INVERT), and sometimes with 2, but, can we use fewer?

The TRICK is to OR the ANDs of all input combinations that generate an output of "1". You don't need the OR gate if only one input combination results in a "1".

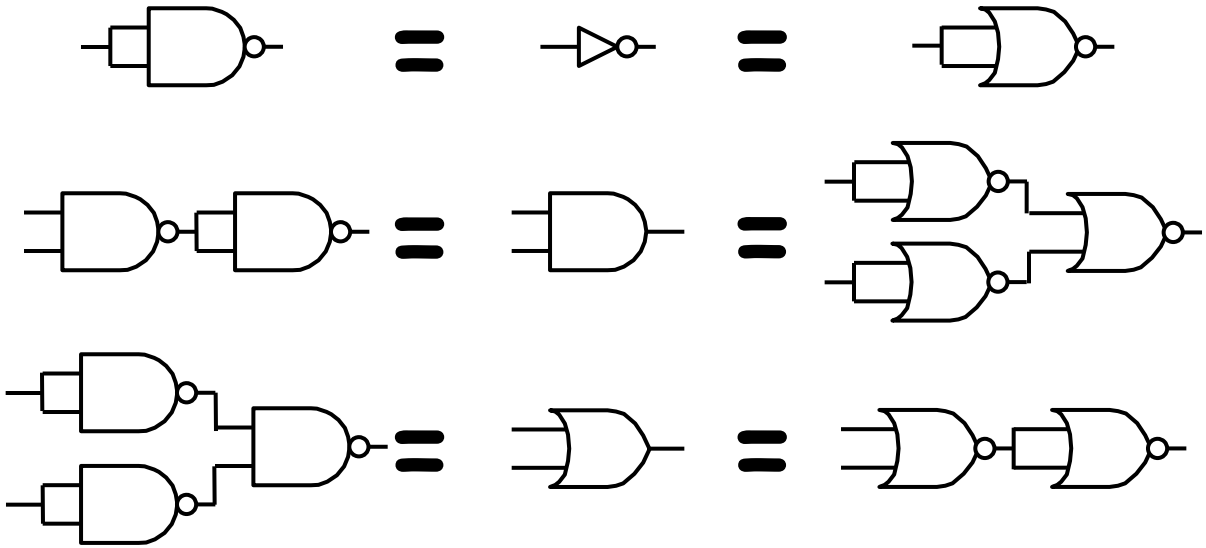
You need Inverters to handle input combinations involving "0"s, ANDs, and ORs.

ONE WILL DO!

A UNIVERSAL gate is one that can be used to implement *ANY* COMBINATIONAL FUNCTION. There are many UNIVERSAL gates, but not all gates are UNIVERSAL

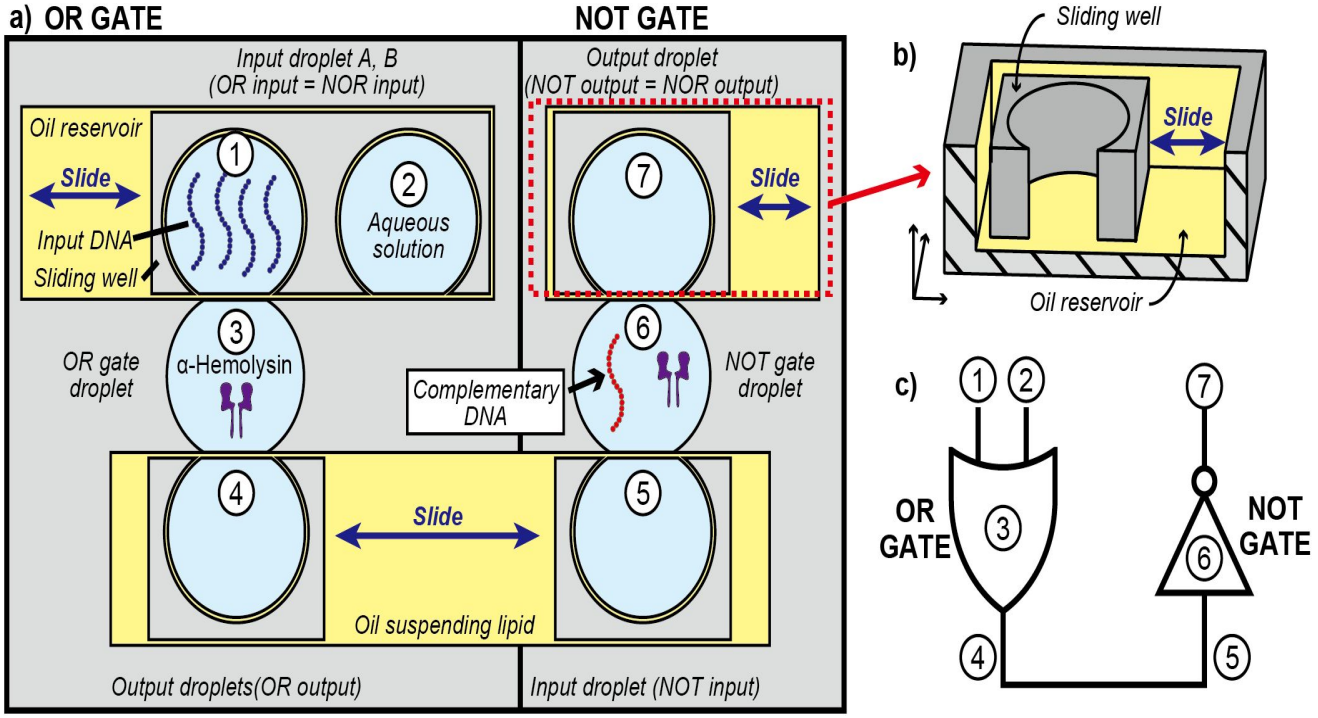
Q: What is a COMBINATIONAL FUNCTION?
A: Any function that can be written as a truth table.

NANDs and NORs are UNIVERSAL!



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

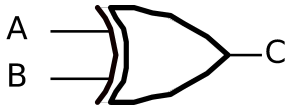
A Few More Gates



- Composing gates for high fan-in
- SOP logic synthesis
- MUX logic

Stupid Gate Tricks

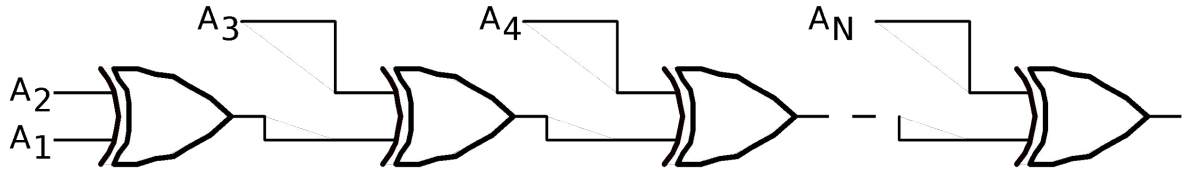
Suppose we have some 2-input XOR gates:



t_{pd} = 1 nS

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

And we want an N-input XOR:



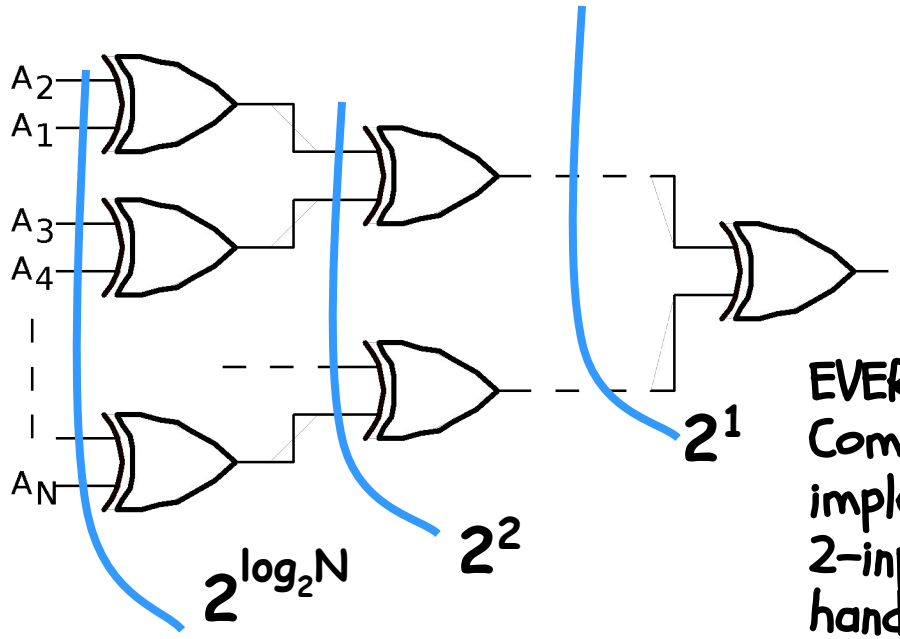
output = 1 iff number of "1"s input is ODD ("PARITY")

t_{pd} = N nS -- WORST CASE.

Can we compute an N-input XOR faster?

I Think That I Shall Never See

a Gate Lovely as a ...



EVERY N-Input Combinational function be implemented using only 2-input gates? But, it's handy to have gates with more than 2-inputs when needed.

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

Signal propagation takes O(log N) gate delays.

A SYSTEMATIC DESIGN APPROACH

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

-it's systematic!
-it works!
-it's easy!
-we get to go home!

1) Write the functional spec as a truth table
2) Write down a Boolean expression for every '1' in the output

```
Y = (!C && !B && A) || (!C && B && A)
  || (C && B && !A) || (C && B && A)
```

3) Wire up the ideal gates, replace them with equivalent realizable gates, call it a day, and go home!

This approach will always give us logic expressions in a particular form:

SUM-OF-PRODUCTS

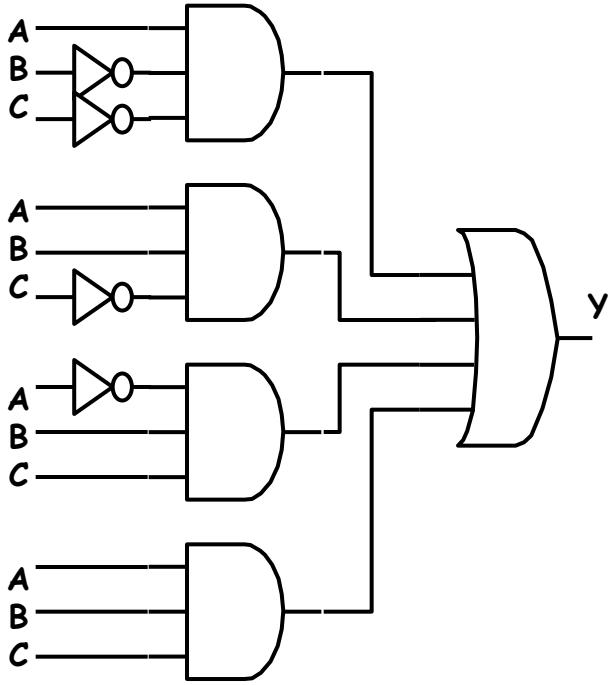
Straightforward Synthesis

We can implement

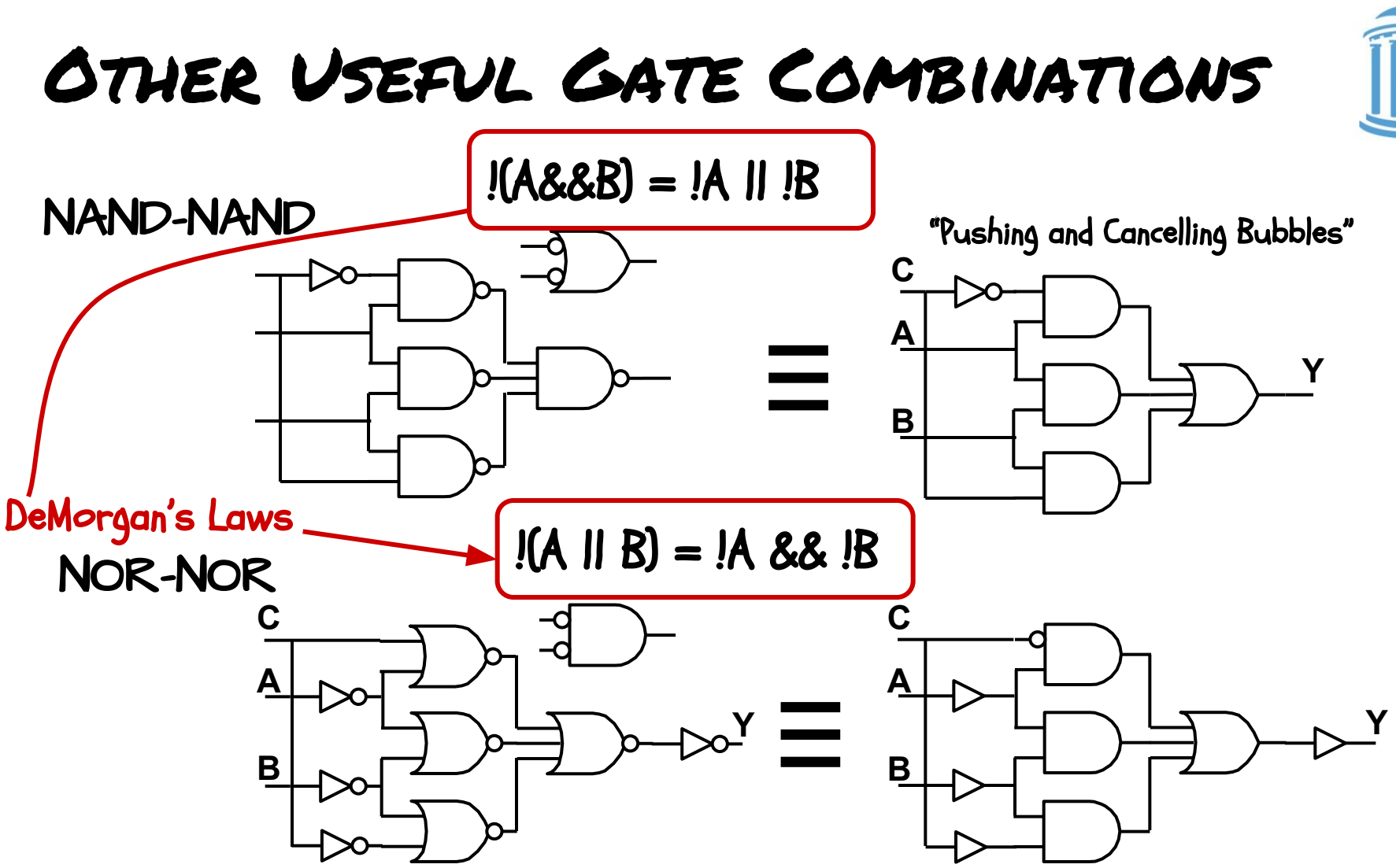
SUM-OF-PRODUCTS

with just 3 levels of logic.

INVERTERS/AND/OR

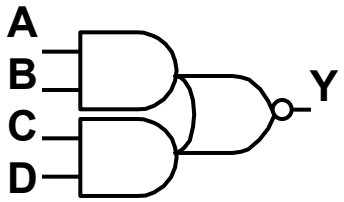


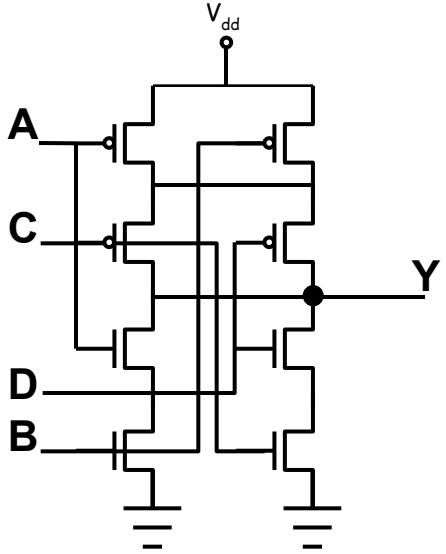
Other Useful Gate Combinations



Other Useful CMOS Gates

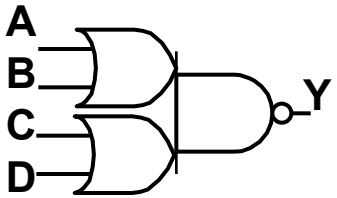
AOI (AND-OR-INVERT)

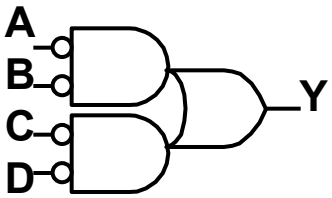


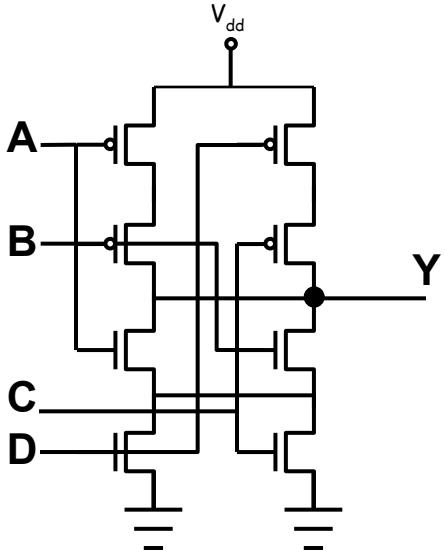


AOI and OAI structures can be realized as a single CMOS gate. However, their function is equivalent to 3 levels of logic.

OAI (OR-AND-INVERT)



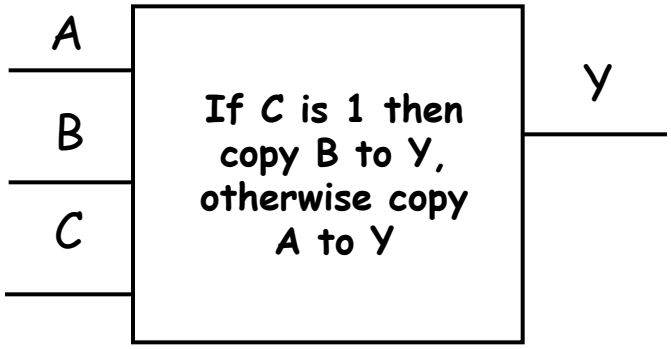




An OAI's DeMorgan equivalent is usually easier to think about.

Another Interesting 3-Input Gate

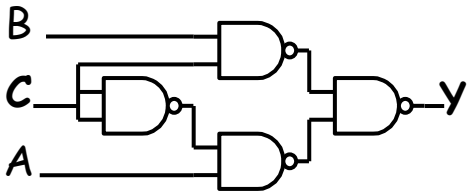
Based on C, select the A or B input to be copied to Y.



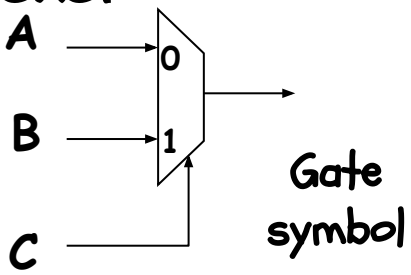
Truth Table

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

2-input Multiplexer



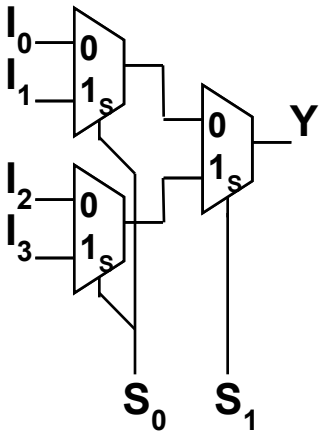
schematic

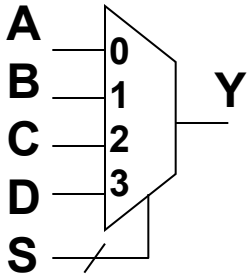


Gate symbol

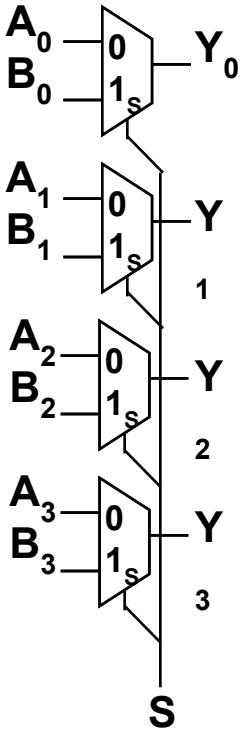
MUX Compositions and Shortcuts

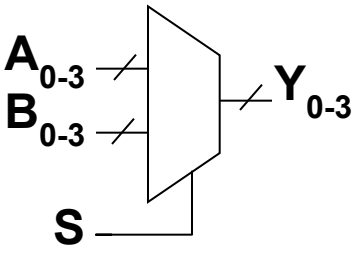
A 4-input Mux
(implemented as a tree)





A 4-bit wide 2-input Mux



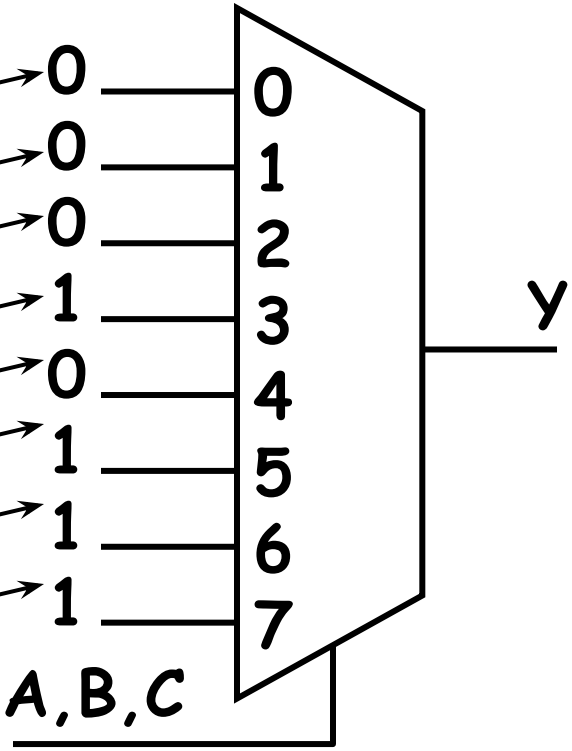


MUX Function Synthesis

Consider implementation of some arbitrary Combinational function, F(A,B,C)… using a MULTIPLEXER as the only circuit element:

Mux Logic: An example "configurable" logic element

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

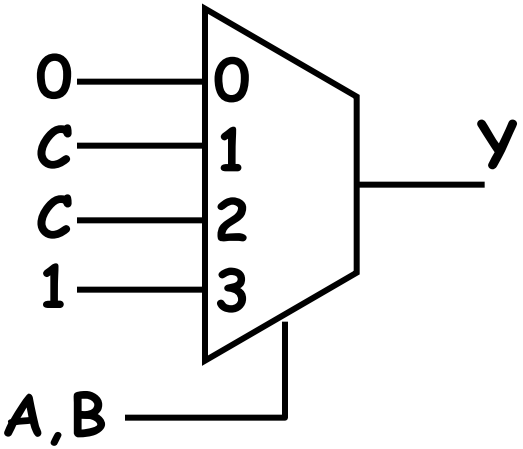


MUX Logic Tricks

We can apply certain optimizations to MUX Function synthesis

Desired Logic Function

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



- Largely by inspection or exhaustive search

- Build *any* N-input gate with an N-1 input MUX & one inverter

Next Time

Binary Circuits that:

- ADD
- SUBTRACT
- SHIFT