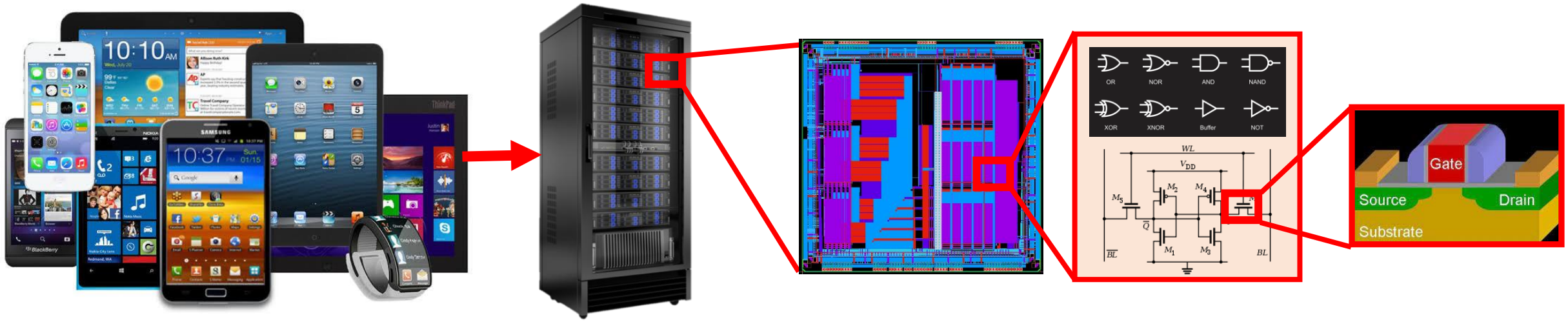


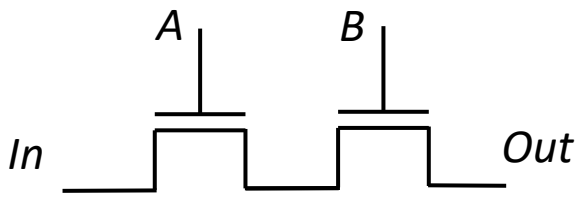
ESC201: Introduction to Electronics

Module 6: Digital Circuits

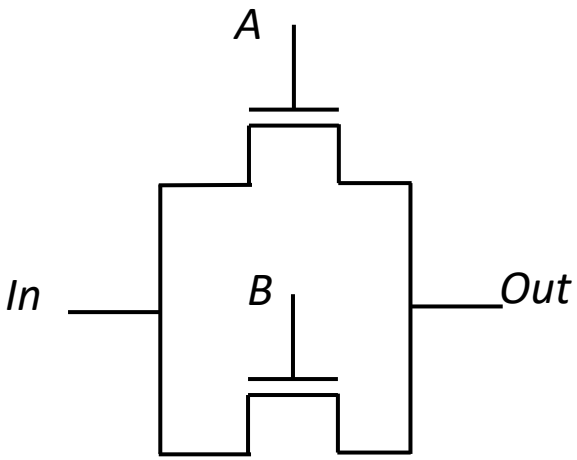
Dr. Shubham Sahay,
Associate Professor,
Department of Electrical Engineering,
IIT Kanpur

Combining Switches

N-MOSFET (positive) switches

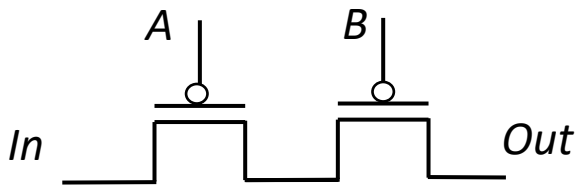


Out transparent to *In* for $A \cdot B = 1$

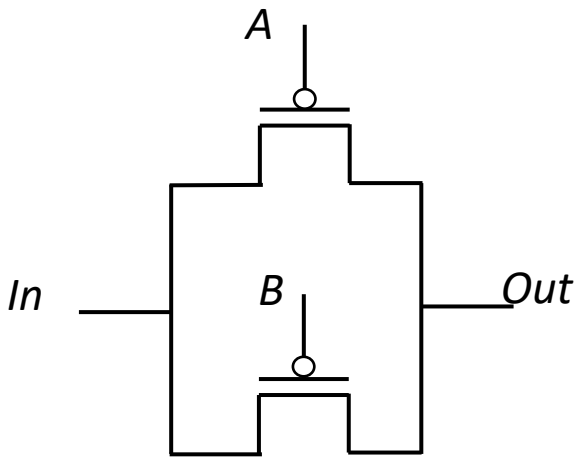


Out transparent to *In* for $A + B = 1$

P-MOSFET (negative) switches

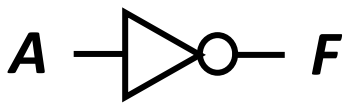


Out transparent to *In* for $\overline{A} \cdot \overline{B} = 1$



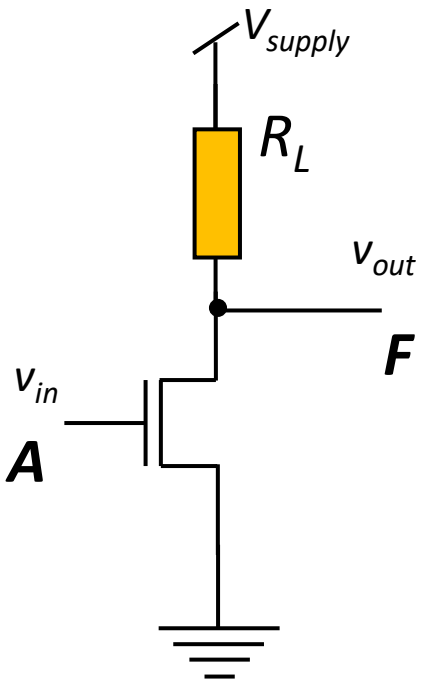
Out transparent to *In* for $\overline{A} + \overline{B} = 1$

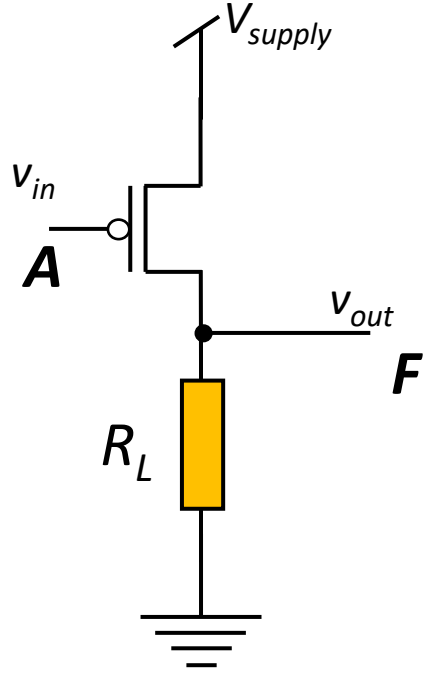
Inverters or NOT Gate

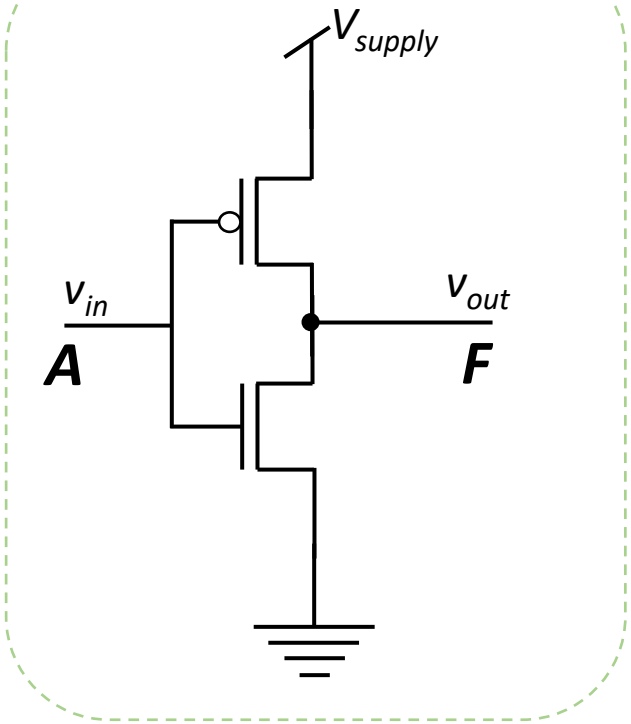


$F = \overline{A}$

A popular solution:
CMOS inverter





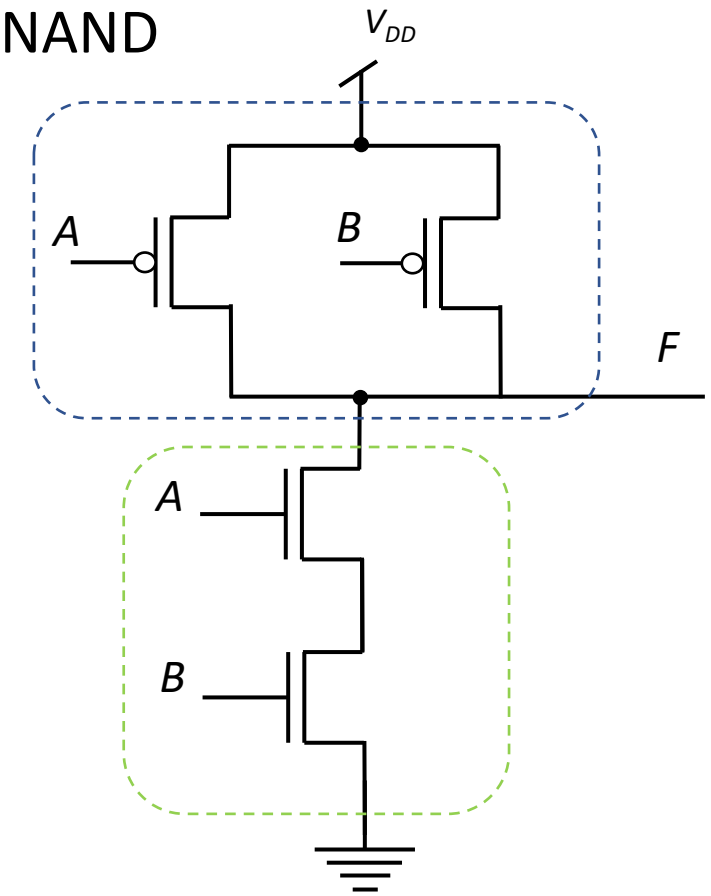


V_{in} and V_{out} are analogue values of input and output voltage

A and *B* are Boolean values of input and output.

Popular Two Input Universal Gates

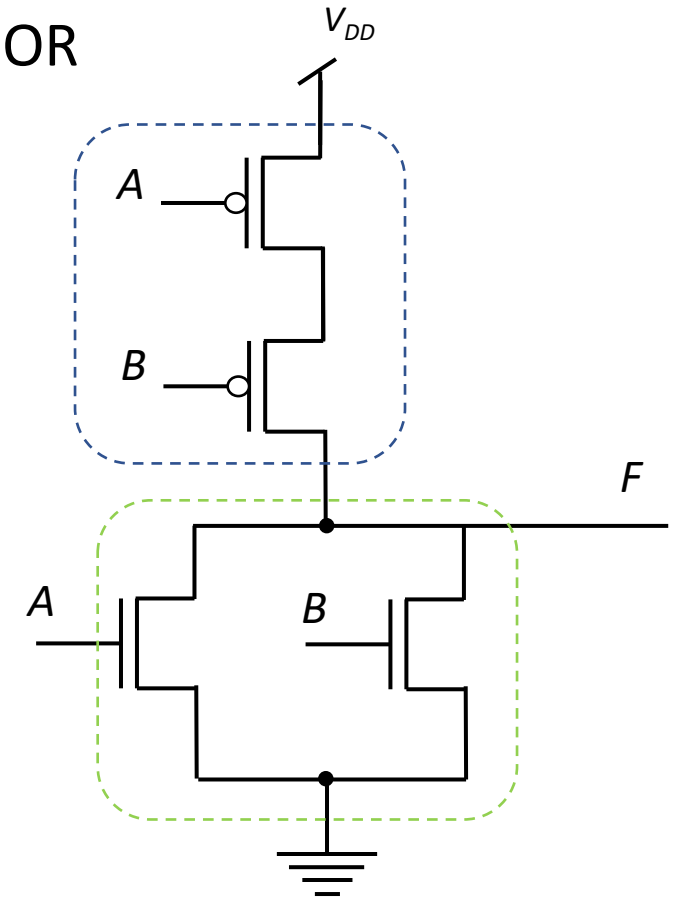
NAND



Two input NAND Gate

$F = \overline{A \cdot B} = \overline{A} + \overline{B}$

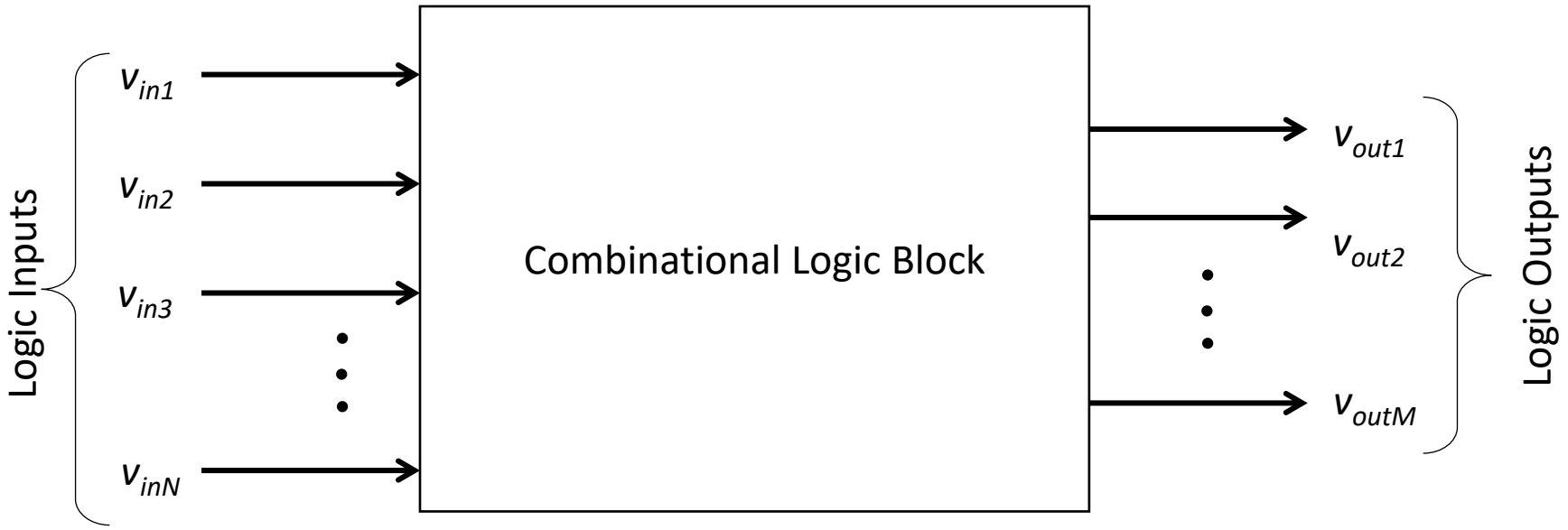
NOR



Two input NOR Gate

$F = \overline{A + B} = \overline{A} \cdot \overline{B}$

Combinational Logic



$v_{outi} = f_i(v_{in1}, v_{in2}, \ldots, v_{inN})$ for $i = 1$ to M

Here the f_i's are Boolean functions

The functions are typically built with logic gates

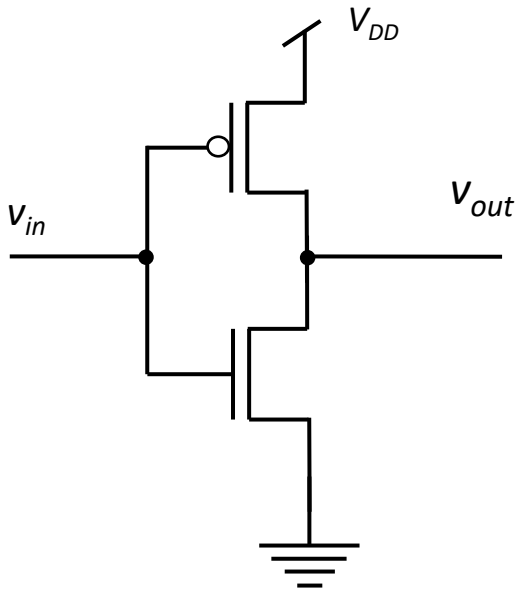
Any circuit that implements a combinational logic is a combinational circuit.

There are many popular methods to implement Combinational circuits

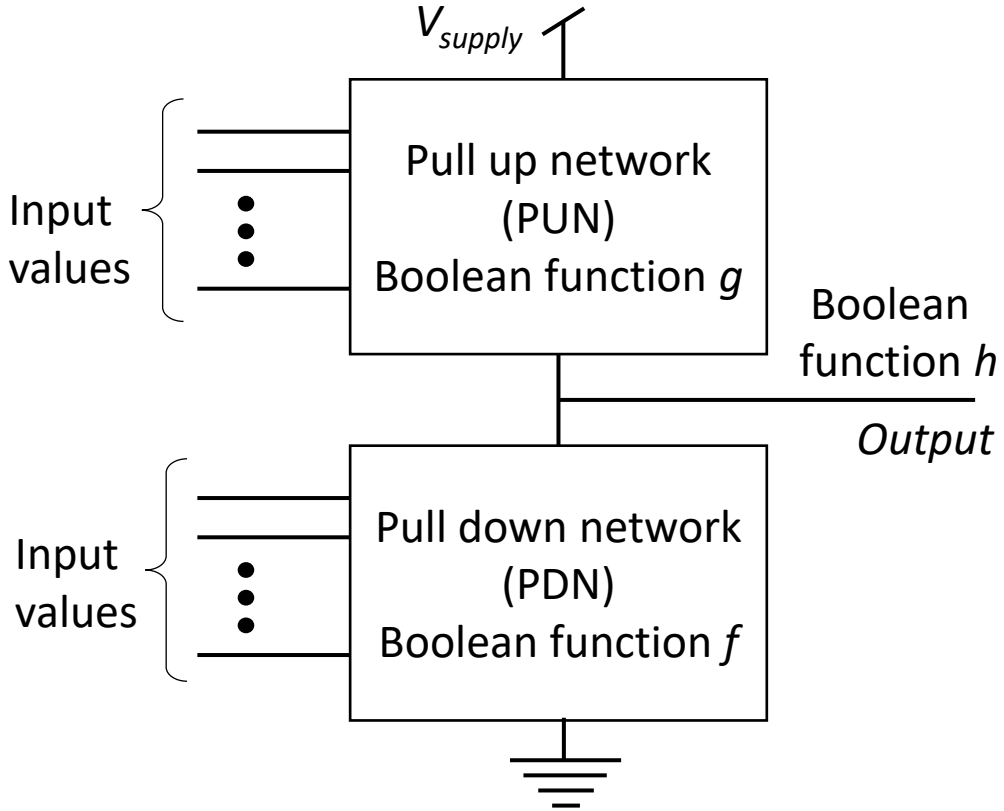
- Complementary CMOS design
- Pseudo n-MOS or Pseudo p-MOS design
- Pass-transistor logic design
- Dynamic circuit design
- …

- Spintronic circuits
- Neuromorphic Computing

CMOS Inverter



Complementary CMOS Circuit



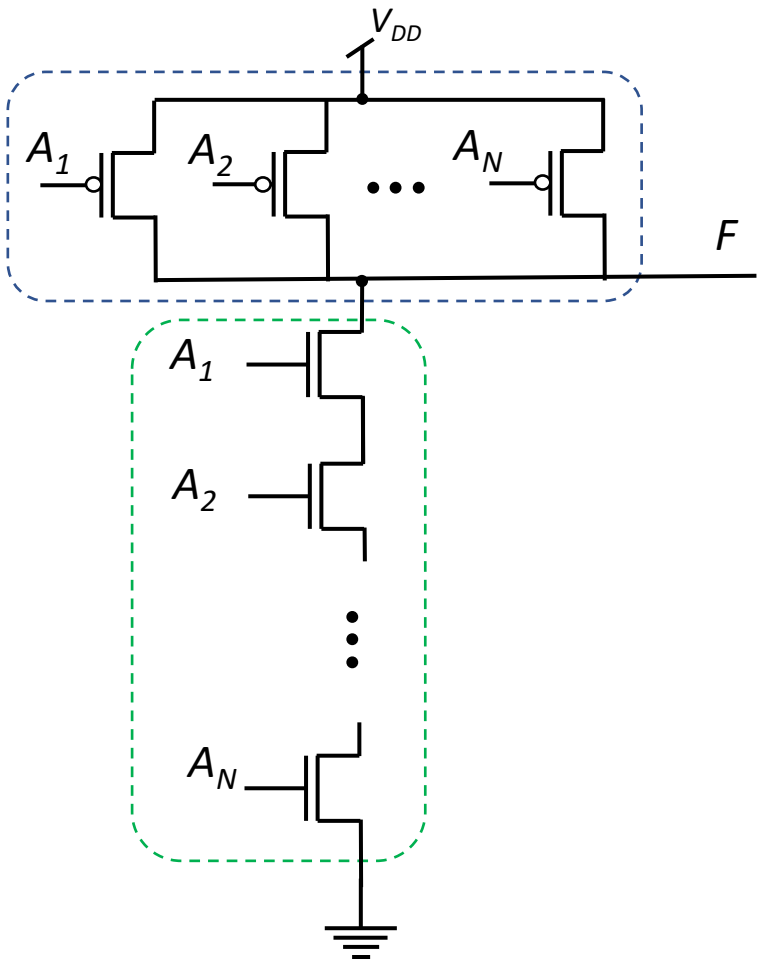
Boolean functions g and f are complements of each other for all input combinations.

Why? Because you don't want V_{DD} and GND to 'fight' over the output node.

Boolean function $h = g = \overline{f}$

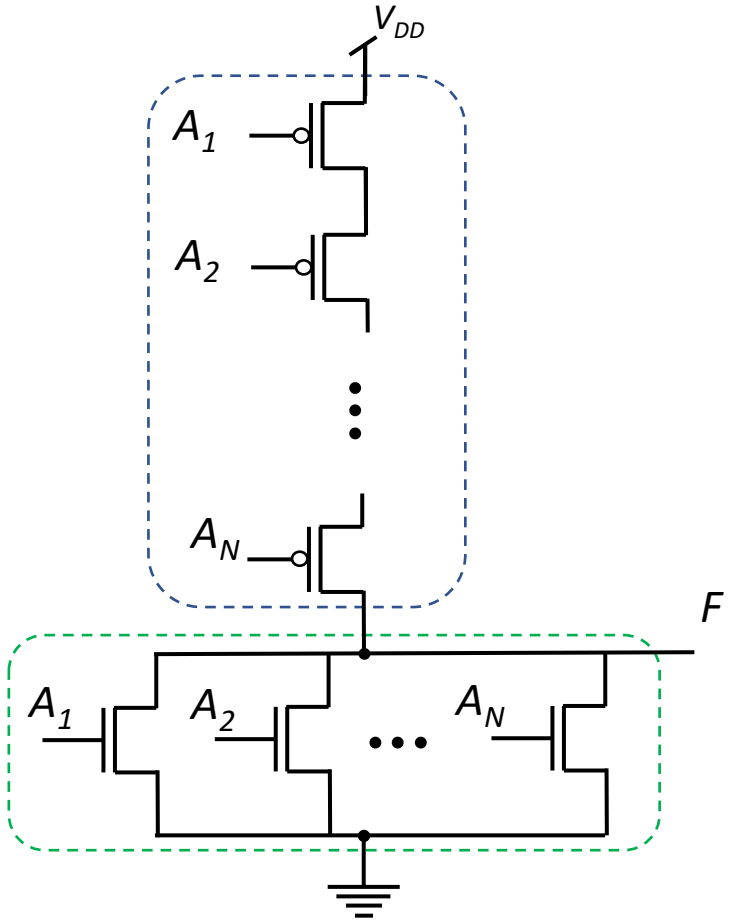
Multi Input NAND or NOR

N input NAND Gate



$$F = \overline{A_1 \cdot A_2 \cdot \ldots \cdot A_N} = \overline{A_1} + \overline{A_1} + \overline{\ldots} + \overline{A_N}$$

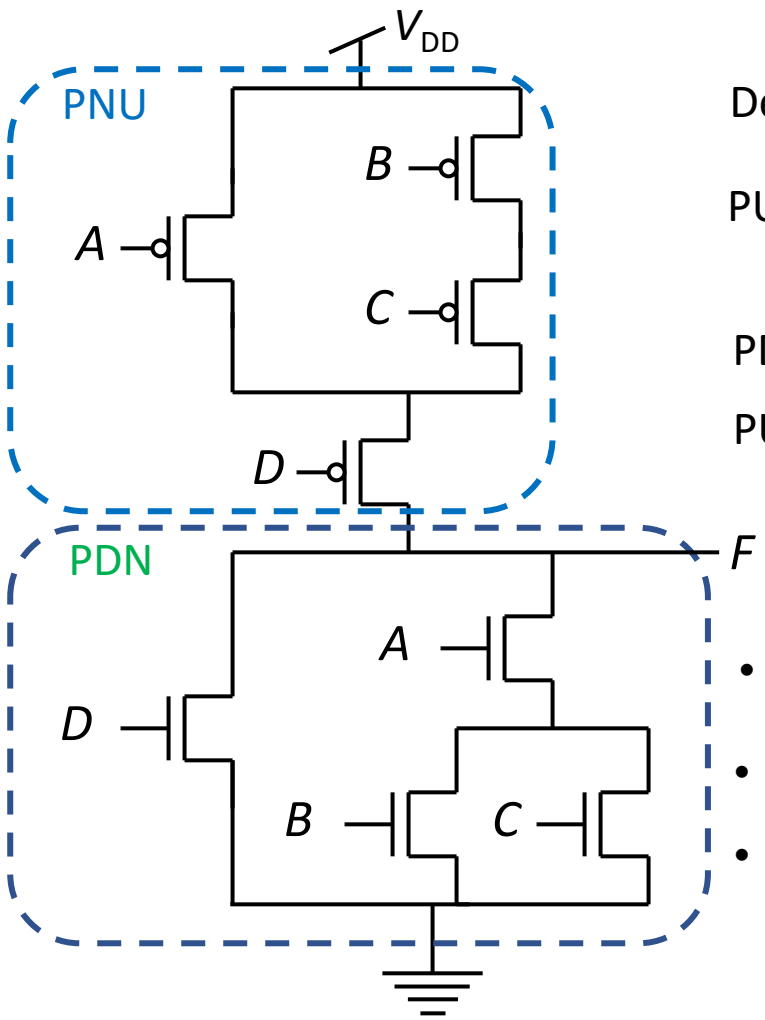
N input NOR Gate



$$F = \overline{A_1 + A_2 + \ldots + A_N} = \overline{A_1} \cdot \overline{A_2} \cdot \ldots \cdot \overline{A_N}$$

Example

Implement the function $F = \overline{D + A \cdot (B + C)}$ as a complementary CMOS circuit



Design PDN function $g = F = \overline{D} + A \cdot (B + C)$

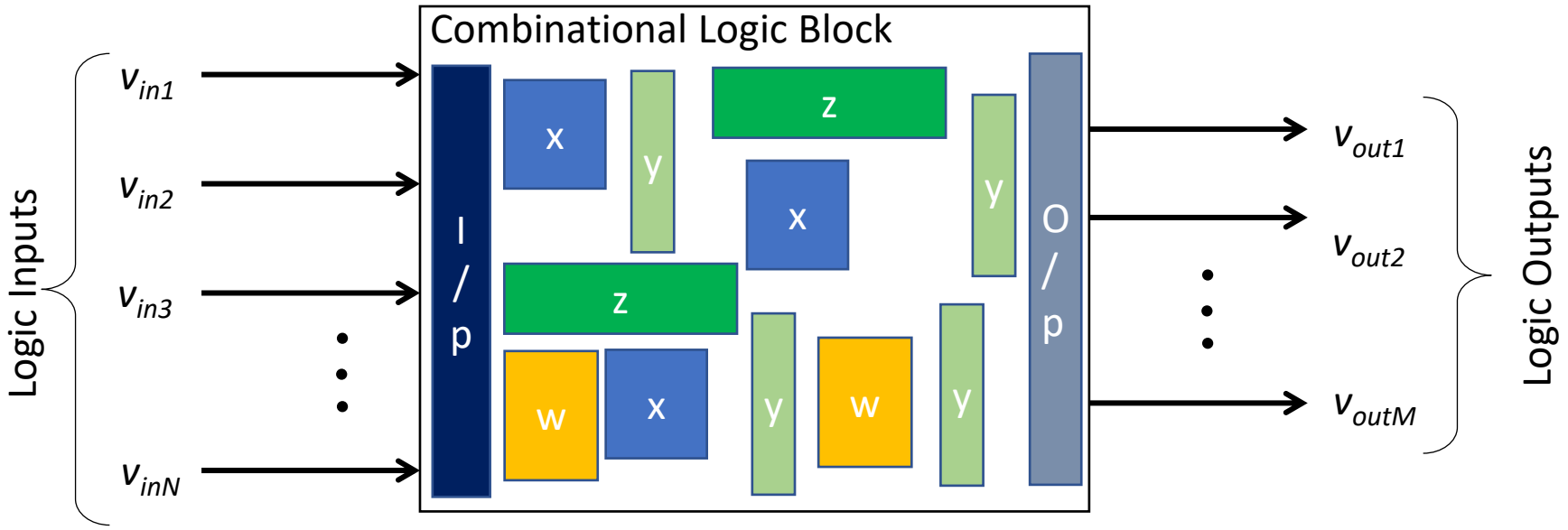
PUN function $h = F = D \cdot \overline{(A + \overline{B} \cdot \overline{C})}$

PDN are nMOSFET → Positive switches

PUN are pMOSFET → Negative switches

- PUN function is complement of PDN function
- What is in series in PDN is in parallel in PUN.
- What is in parallel in PDN is in series in PUN.

Useful Circuit Blocks

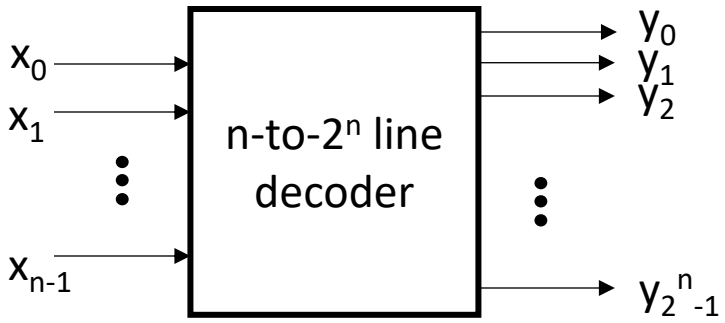


Build the larger circuit by using many smaller combinational logic functional blocks.

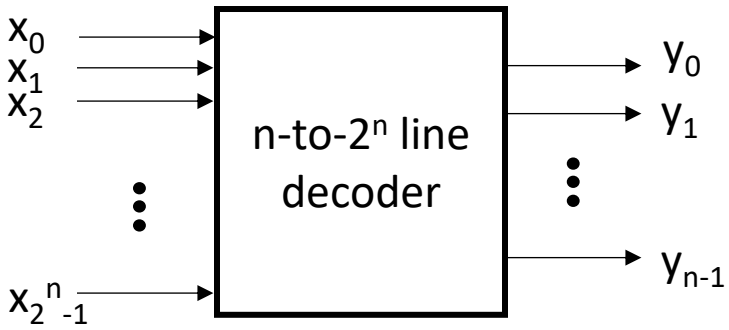
MUX, DeMUX, Encoder, Decoder, … adder, subtractor, multiplier, … and so on.

Only in special situations will one want to re-design to optimise the integrated block.

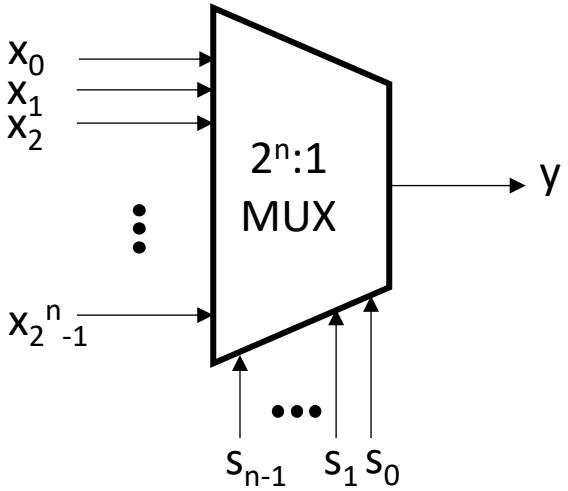
Decoder



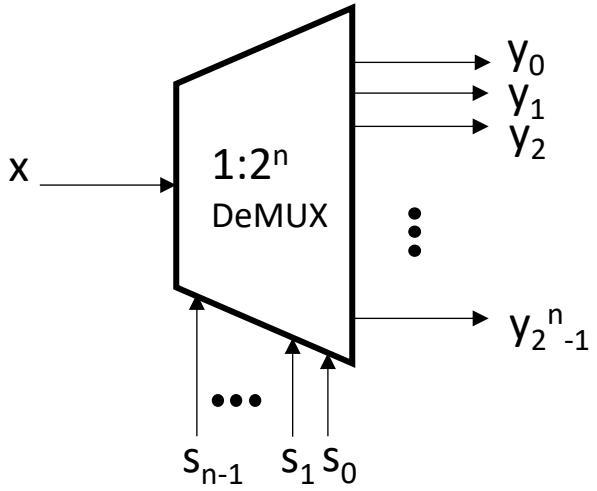
Encoder



Multiplexer (MUX)

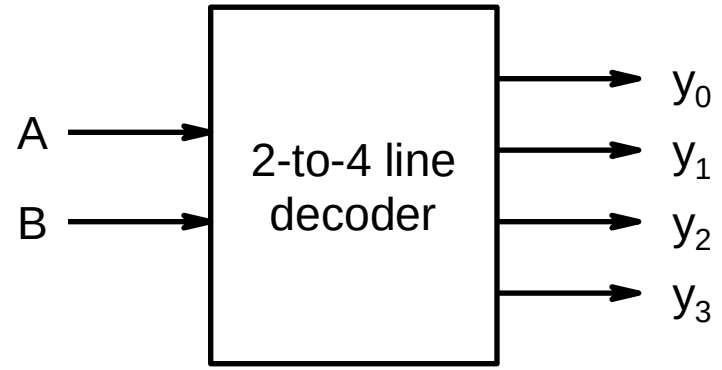


Demultiplexer (DeMUX)



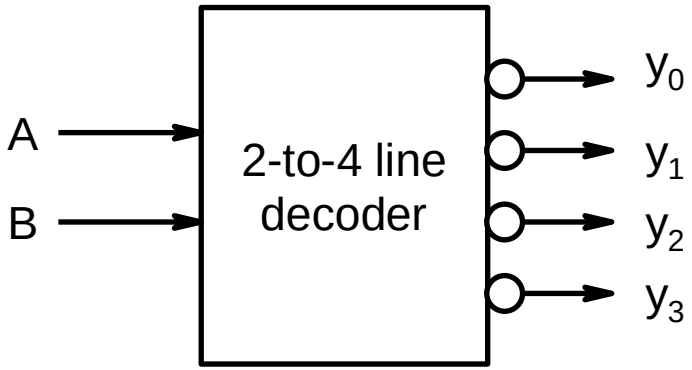
2-to-4 Line Decoder

Maps a smaller number of inputs to a larger set of outputs in general



B	A	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

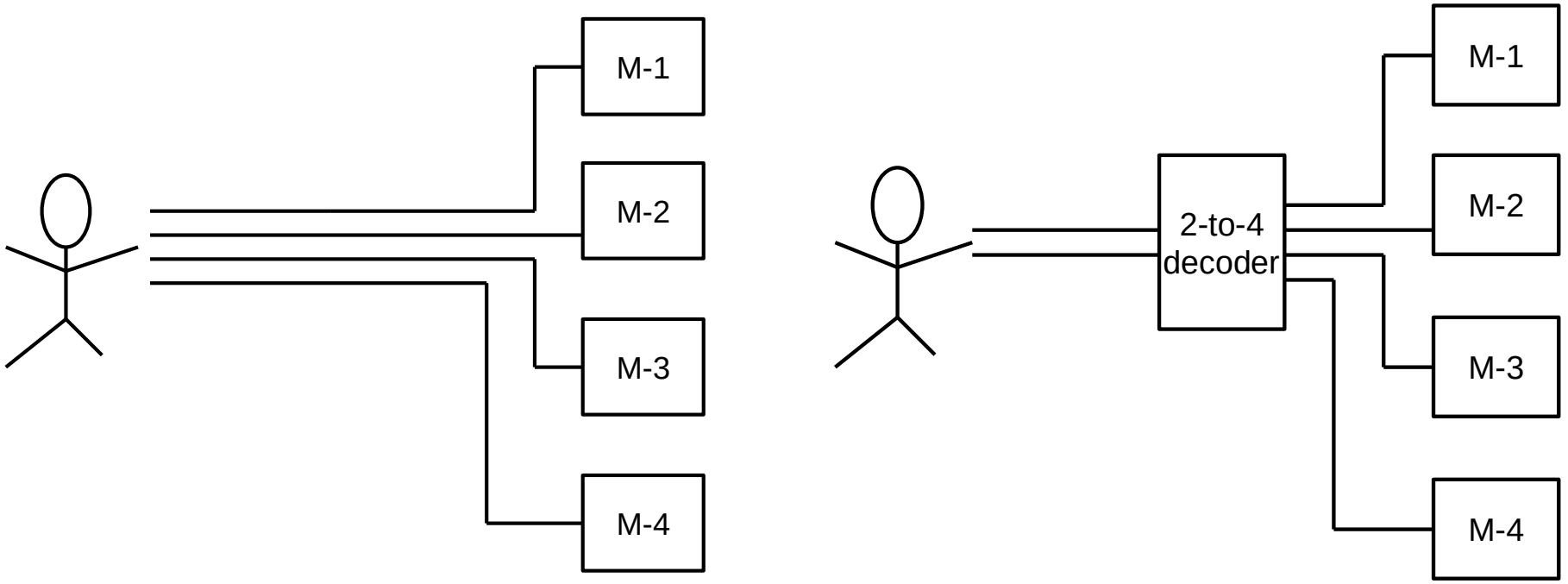
Active High



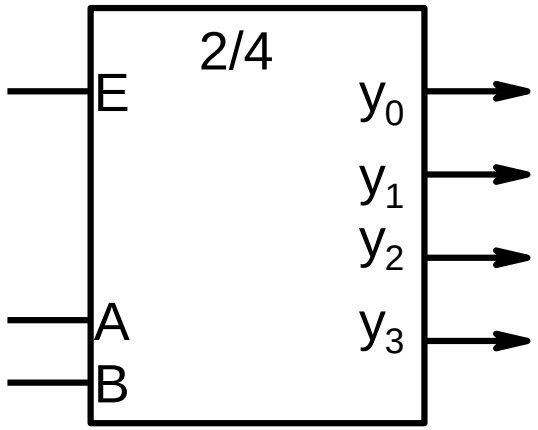
b	a	Y_0	Y_1	Y_2	Y_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Active Low

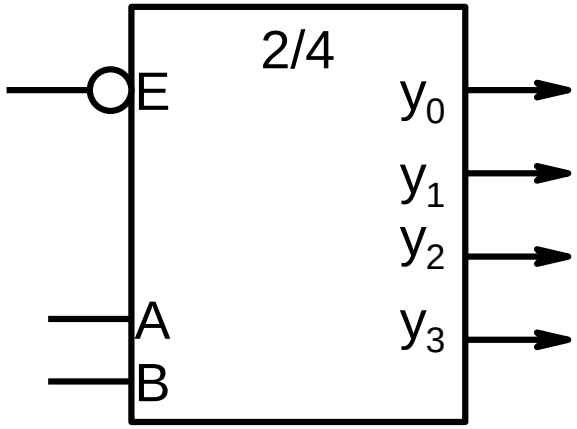
Decoder Interprets Coded Data



Decoder with Enable Input

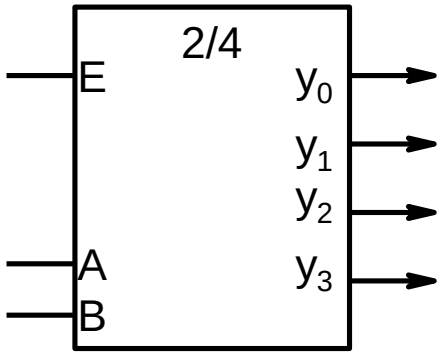


E	B	A	Y_0	Y_1	Y_2	Y_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1



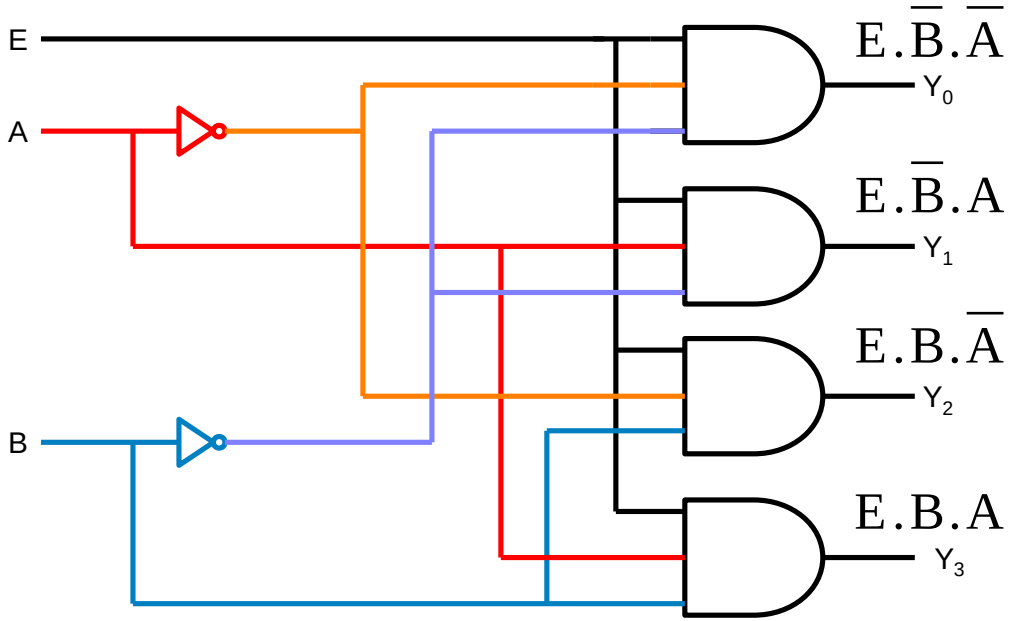
E	B	A	Y_0	Y_1	Y_2	Y_3
1	x	x	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1

Decoder Gate Implementation



E	B	A	Y_0	Y_1	Y_2	Y_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$$Y_0 = E.\overline{B}.\overline{A}\ ; Y_1 = E.\overline{B}.A ; Y_2 = E.B.\overline{A}\ ; Y_3 = E.B.A$$



A *n* to 2^n decoder is a minterm generator

By selecting the min-terms, one may implement a truth table function!