CHAPTER II

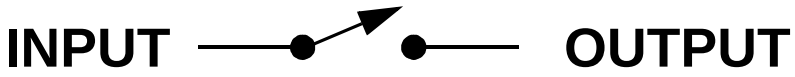
SWITCH NETWORKS AND SWITCH DESIGN

SWITCH NETWORKS

BASIC IDEAL SWITCH

- Simplest structure in a computing system is a switch

IDEAL SWITCH

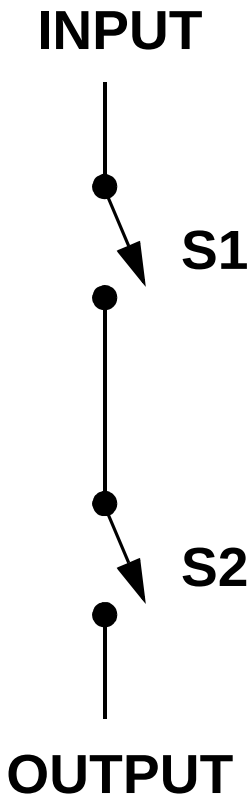


- Path exists between INPUT and OUTPUT if Switch is CLOSED or ON
- Path does not exist between INPUT and OUTPUT if SWITCH is OPEN or OFF

SWITCH NETWORKS

SWITCHES IN SERIES

SWITCHES IN SERIES



Truth Table

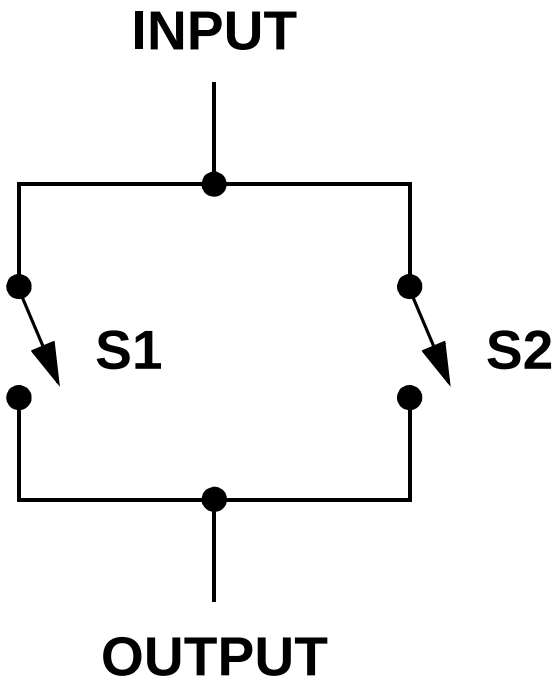
S1	S2	PATH?
OFF	OFF	NO
OFF	ON	NO
ON	OFF	NO
ON	ON	YES

- AND configuration

SWITCH NETWORKS

SWITCHES IN PARALLEL

SWITCHES IN PARALLEL



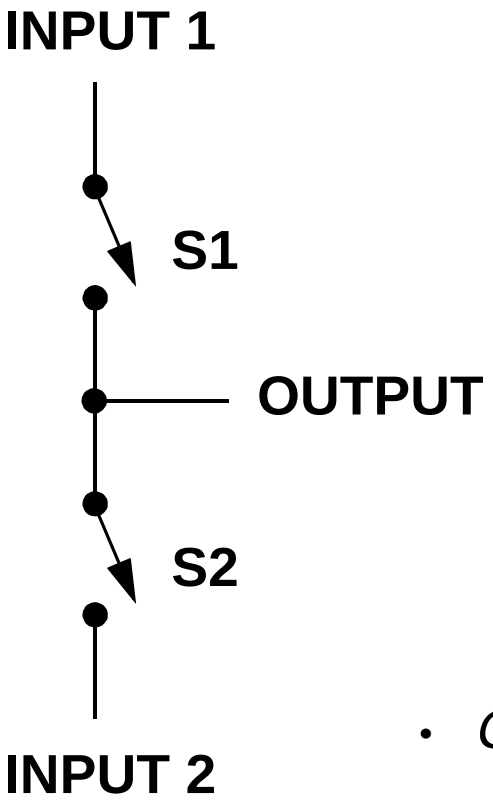
Truth Table

S1	S2	PATH?
OFF	OFF	NO
OFF	ON	YES
ON	OFF	YES
ON	ON	YES

- OR configuration

SWITCH NETWORKS

INPUT SELECTOR



Truth Table

S1	S2	OUTPUT
OFF	OFF	NONE
OFF	ON	INPUT 2
ON	OFF	INPUT 1
ON	ON	UNKNOWN

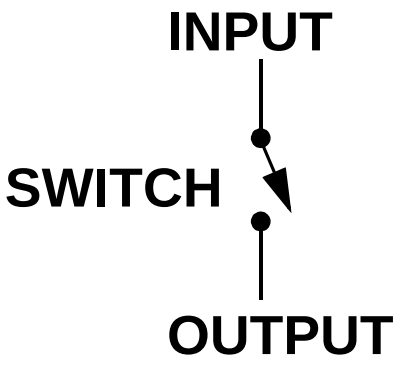
- *Crowbarred* level where logic level is indeterminate. Likely avoid this case.

CMOS

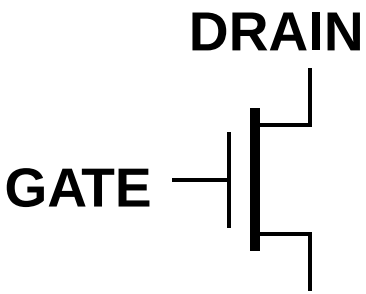
CMOS SWITCHES

- The idea is to use the series and parallel switch configurations to route signals in a desired fashion.
- Unfortunately, it is difficult to implement an ideal switch as given.
- Complementary Metal Oxide Semiconductor (CMOS) devices give us some interesting components.

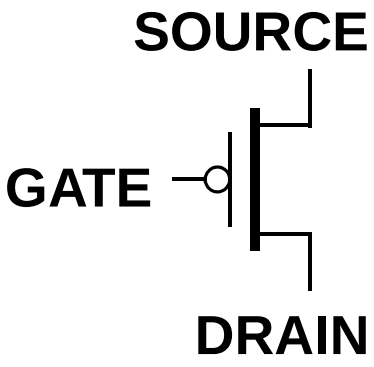
IDEAL SWITCH



nMOS transistor



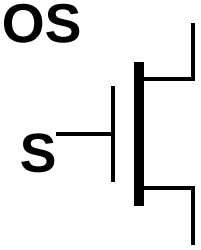
pMOS transistor



CMOS

TRANSFER CHARACTERISTICS

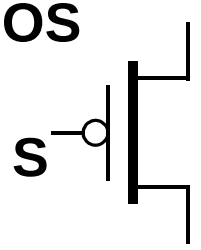
nMOS



S	SWITCH
0	OPEN
1	CLOSED

- nMOS when CLOSED
  - Transmits logic level 0 well
  - Transmits logic level 1 poorly

pMOS



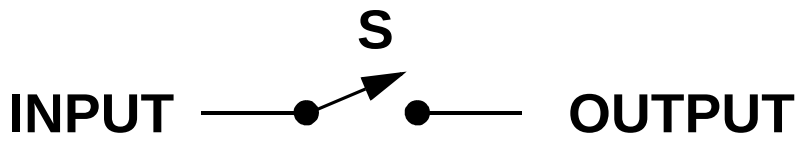
S	SWITCH
0	CLOSED
1	OPEN

- pMOS when CLOSED
  - Transmits logic level 1 well
  - Transmits logic level 0 poorly

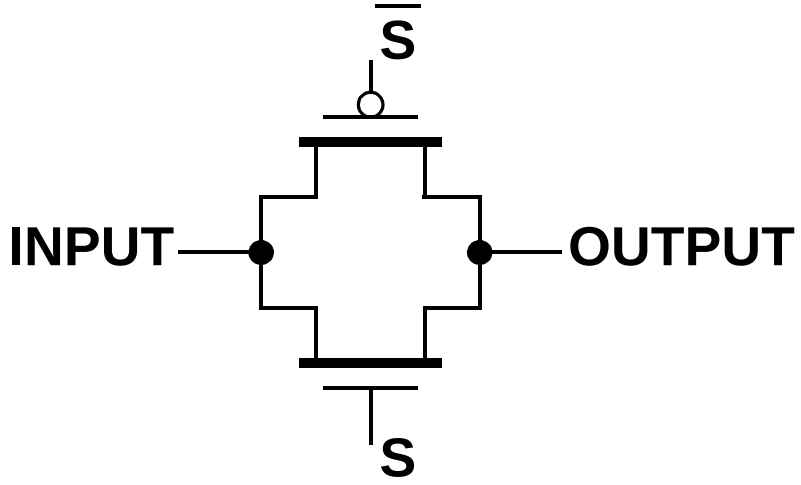
CMOS

TRANSMISSION GATE (1)

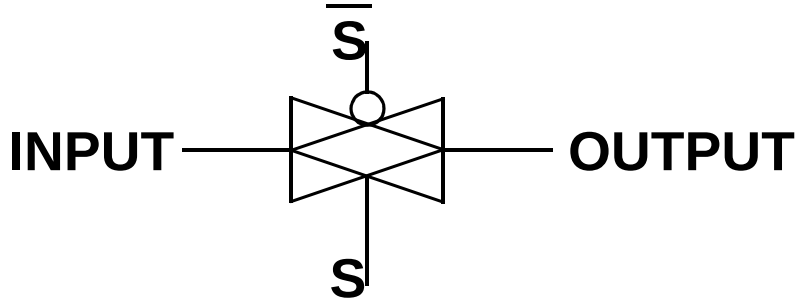
IDEAL SWITCH



CMOS TRANSMISSION GATE (SWITCH)



S	nMOS	pMOS	OUTPUT
0	OFF	OFF	Z
1	ON	ON	INPUT

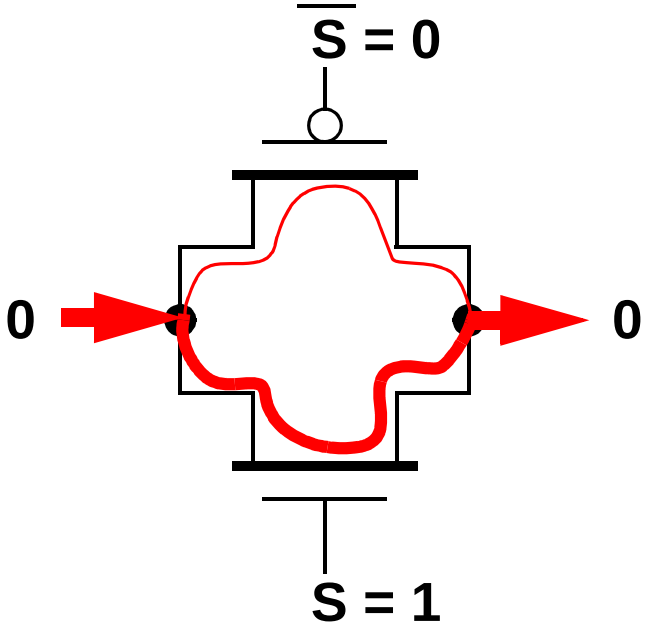


CMOS

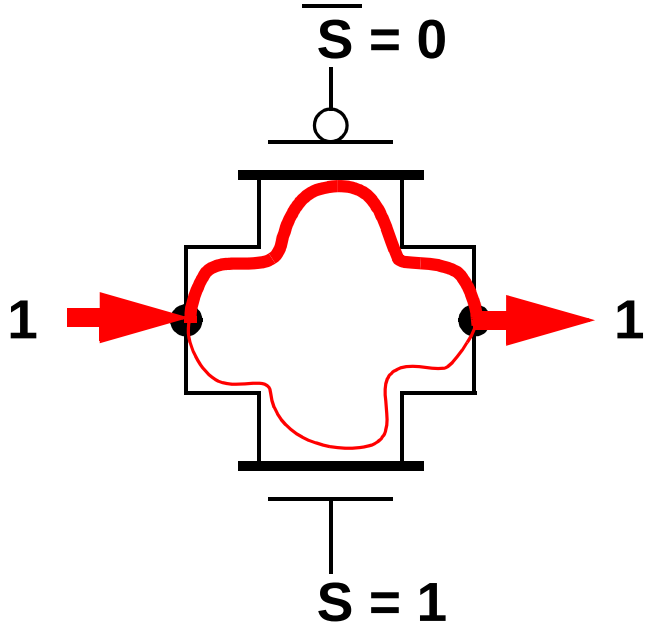
TRANSMISSION GATE (2)

SPLIT OF CURRENT ACROSS A TRANSMISSION GATE FOR LOGIC-0 AND LOGIC-1 INPUT

LOGIC-0 AT INPUT



LOGIC-1 AT INPUT

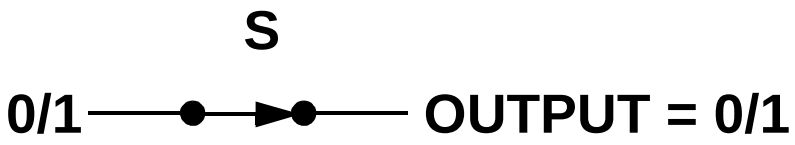


SWITCH NETWORKS

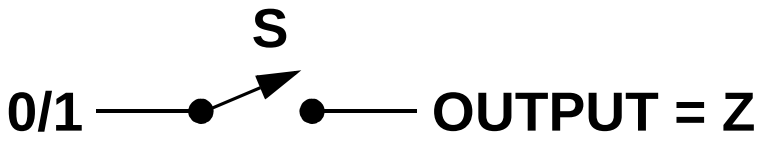
HIGH IMPEDANCE **Z** (1)

- With switches, we can consider three states for an output:
  - Logic-0
  - Logic-1
  - High Impedance **Z**

- Path exists for Logic-0 and Logic-1 when the switch is CLOSED.



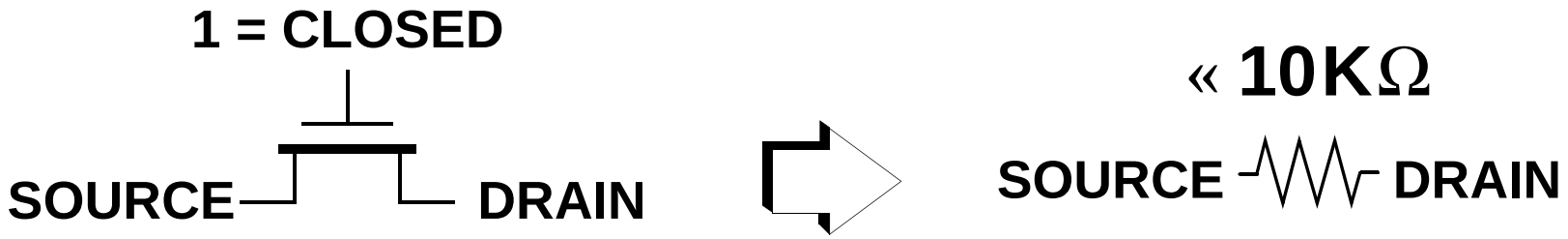
- High impedance is a state where the switch is OPEN.



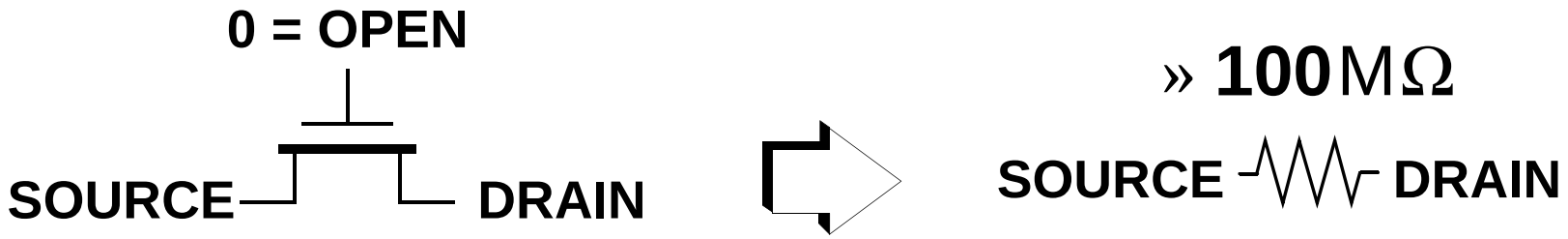
SWITCH NETWORKS

HIGH IMPEDANCE **Z** (2)

- Another way of thinking of switches is as follows
  - Path exists for Logic-0 and Logic-1 when the switch is CLOSED, meaning that the **impedance/resistance is small** enough to allow amply flow of current.



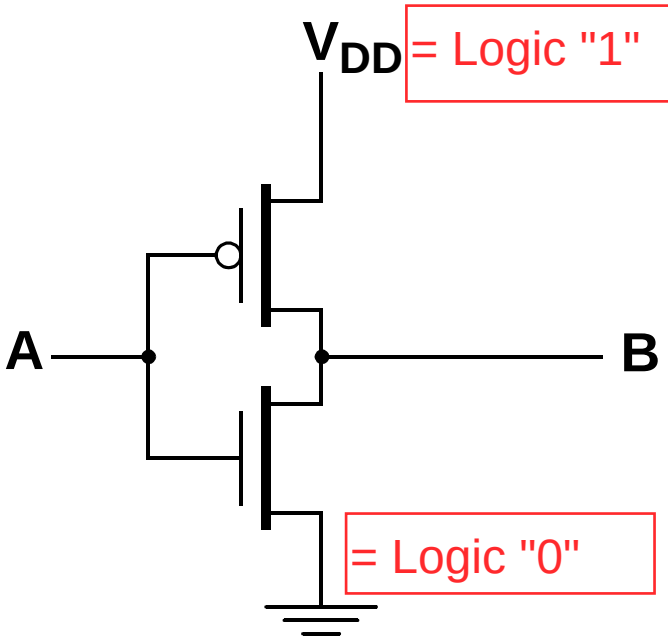
  - High impedance is a state where the switch is OPEN, meaning that the **impedance/resistance is very large** allowing nearly no current flow.



SWITCH NETWORKS

INVERTER (NOT)

$$B = \overline{A}$$



PULL-DOWN		PULL-UP			
A	B	A	B	A	B
0	Z	0	1	0	1
1	0	1	Z	1	0

The pull-down network should be Z "OFF" whenever the output should be "1". It should be ON whenever the output should be "0". Here "ON" implies "0"

The pull-up network should be ON whenever the output should be "1". It should be "Z" (OFF) whenever the output should be "0".

This network inverts the binary input value. N

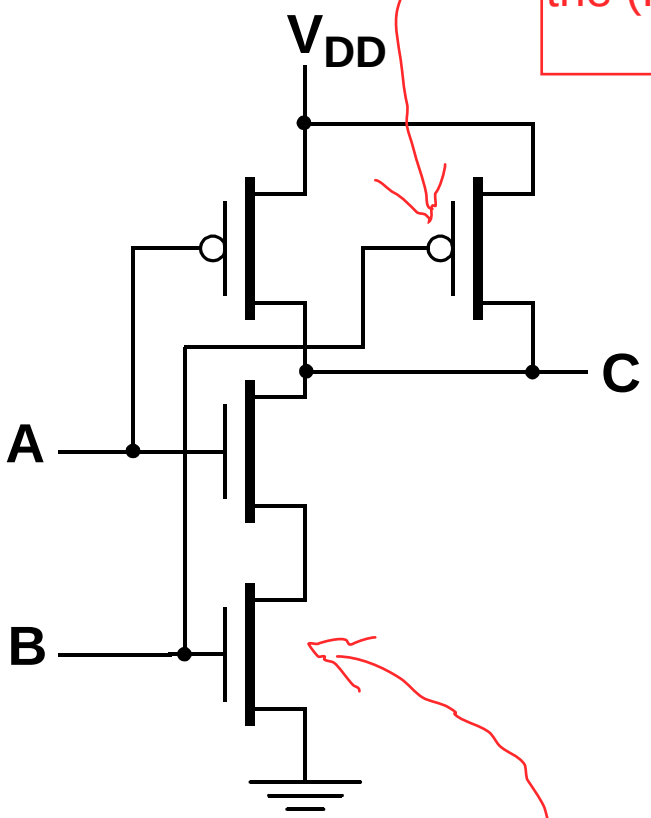
SWITCH NETWORKS

NAND NETWORK

- CMOS
- SWITCH NETWORKS
  - -HIGH IMPEDANCE Z
  - -INVERTER

The "o" on the gate means that "Logic 0" voltage turns the (P-type) transistor "ON".

$$C = \overline{AB}$$



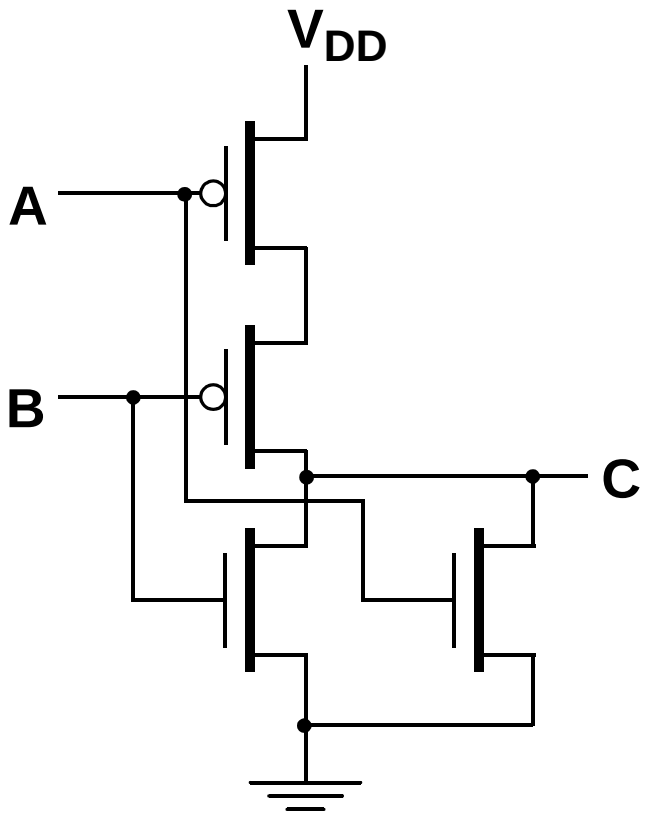
PULL-DOWN			PULL-UP					
A	B	C	A	B	C	→	A	B	C
0	0	Z	0	0	1		0	0	1
0	1	Z	0	1	1		0	1	1
1	0	Z	1	0	1		1	0	1
1	1	0	1	1	Z		1	1	0

When the gate voltage is "Logic 1", the N-type transistor turns ON (no inversion).

SWITCH NETWORKS

NOR NETWORK

- SWITCH NETWORKS
  - -HIGH IMPEDANCE Z
  - -INVERTER
  - -NAND NETWORK



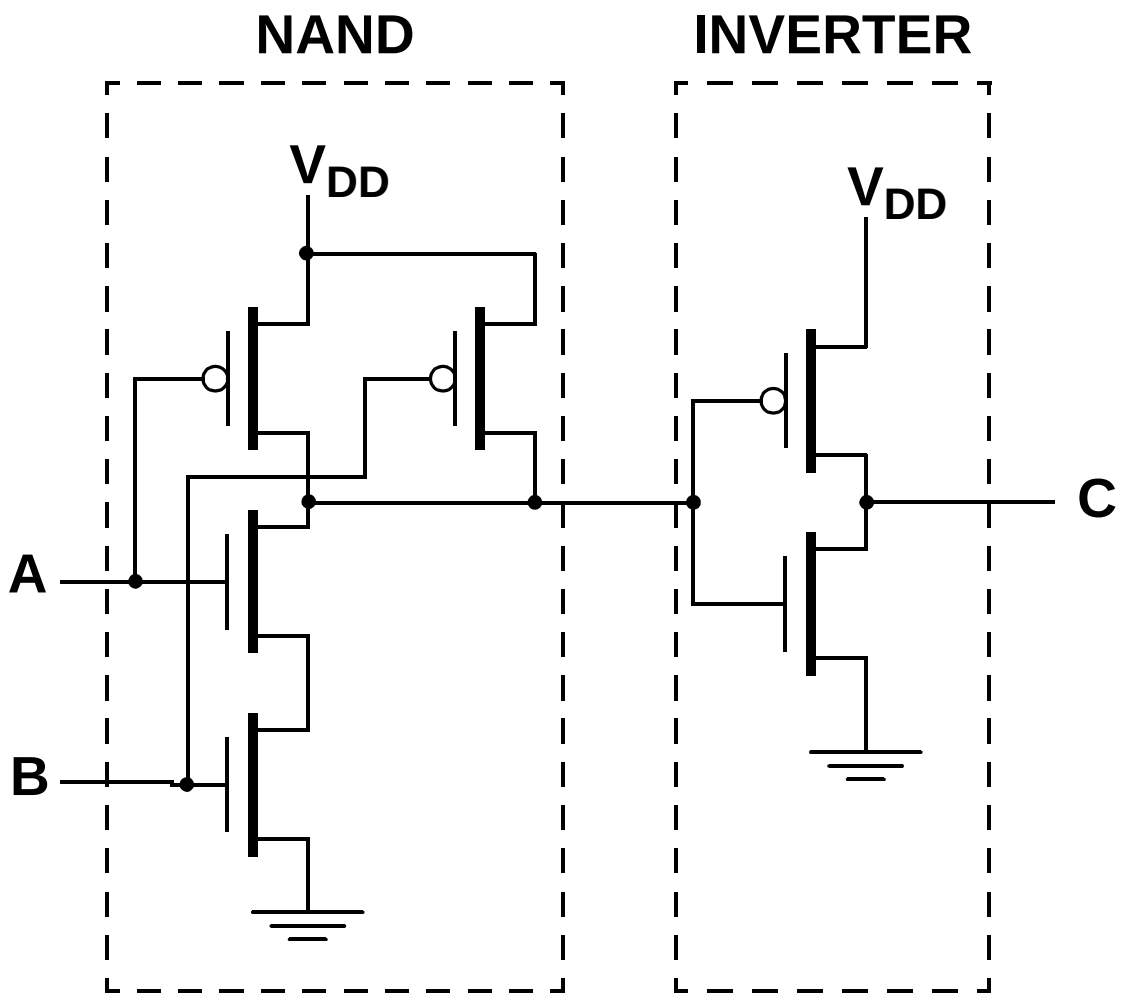
$$C = \overline{A + B}$$

PULL-DOWN			PULL-UP					
A	B	C	A	B	C		A	B	C
0	0	Z	0	0	1		0	0	1
0	1	0	0	1	Z	→	0	1	0
1	0	0	1	0	Z		1	0	0
1	1	0	1	1	Z		1	1	0

SWITCH NETWORKS

AND NETWORK

- SWITCH NETWORKS
  - -INVERTER
  - -NAND NETWORK
  - -NOR NETWORK



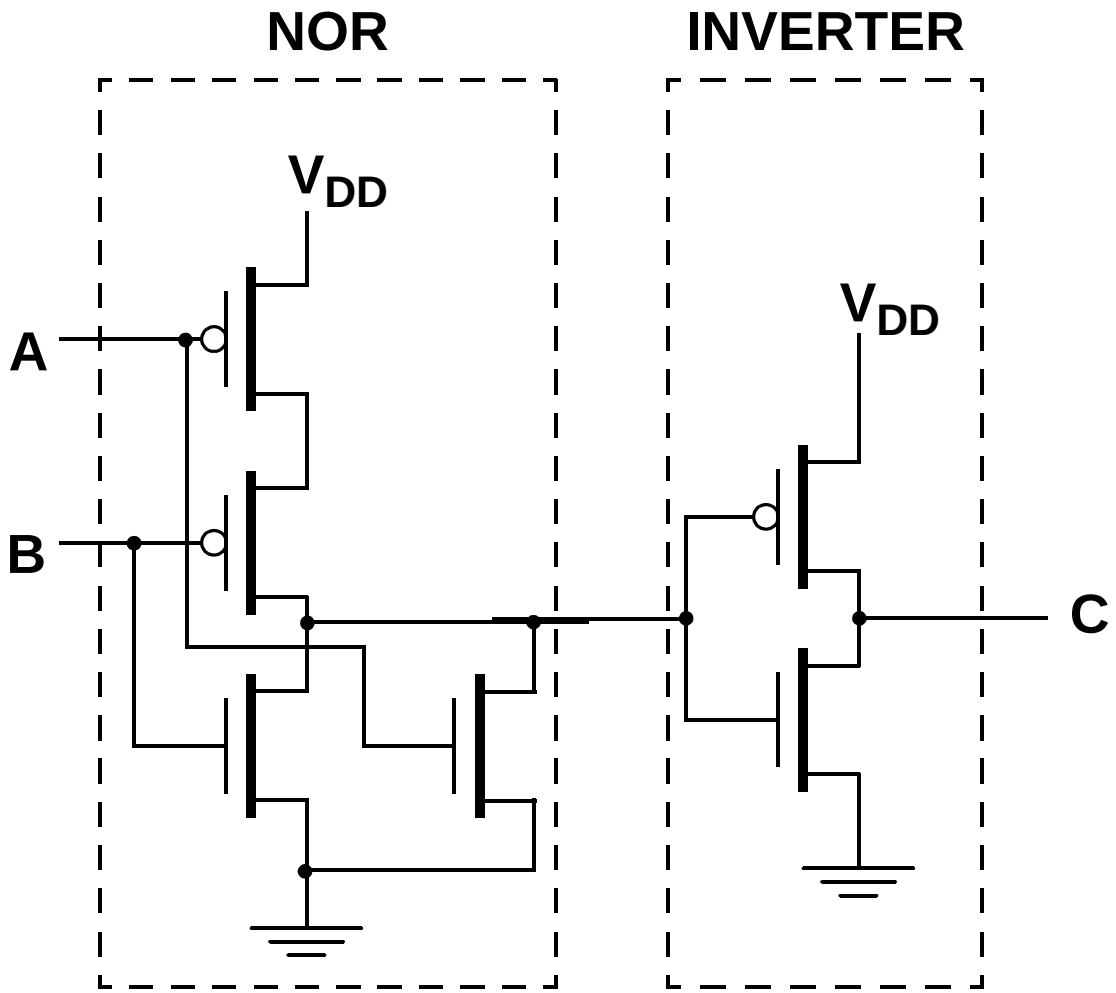
$$C = AB$$

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

SWITCH NETWORKS

OR NETWORK

- SWITCH NETWORKS
  - -NAND NETWORK
  - -NOR NETWORK
  - -AND NETWORK



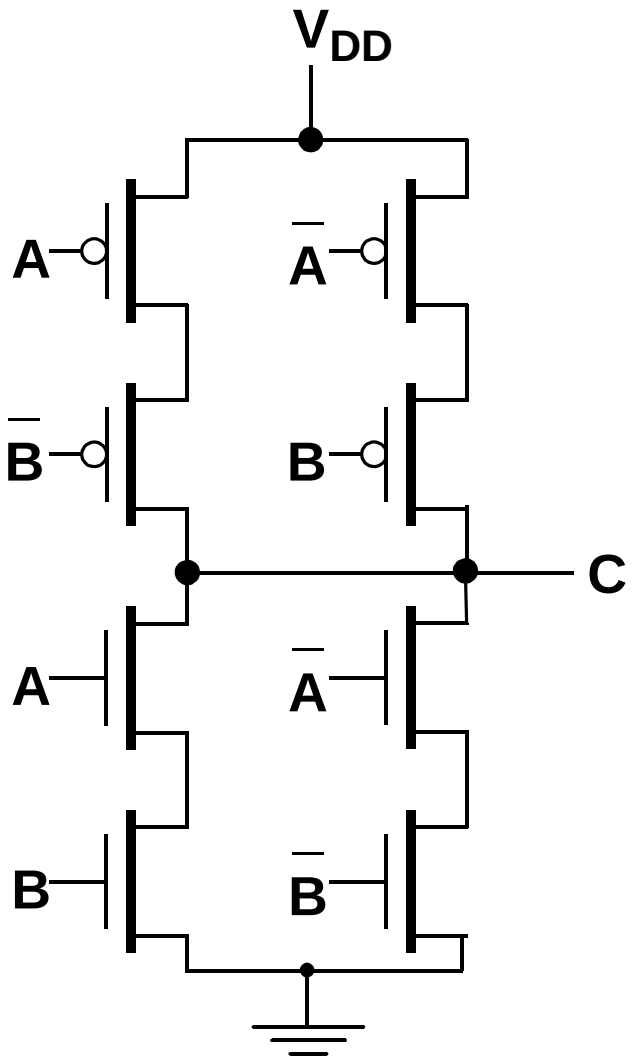
$$C = A + B$$

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

SWITCH NETWORKS

XOR NETWORK

- SWITCH NETWORKS
  - -NOR NETWORK
  - -AND NETWORK
  - -OR NETWORK

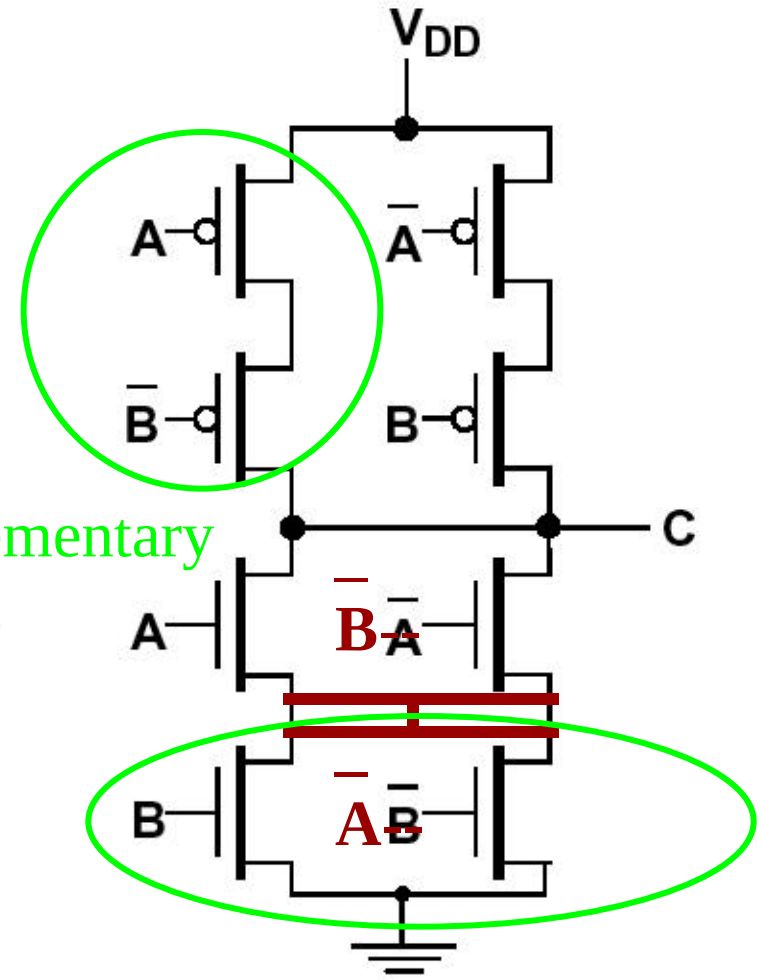


$$C = A\overline{B} + \overline{A}B$$

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

SWITCH NETWORKS

XOR NETWORK



Complementary Circuits

$$C = A\bar{B} + \bar{A}B$$

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

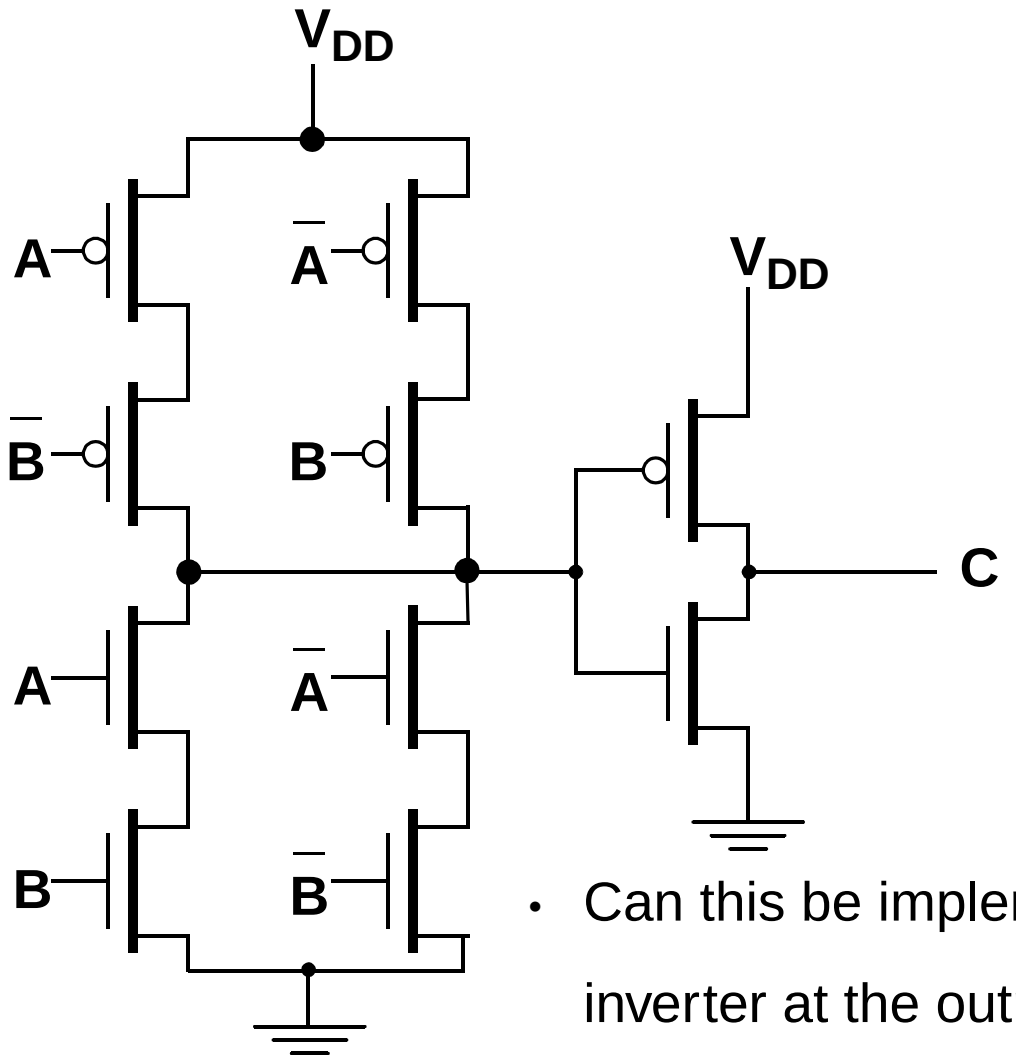
Boolean Equality

AB+A'B' = (A+B')(A'+B)

Because AA' = BB' = 0

SWITCH NETWORKS

XNOR NETWORK



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

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

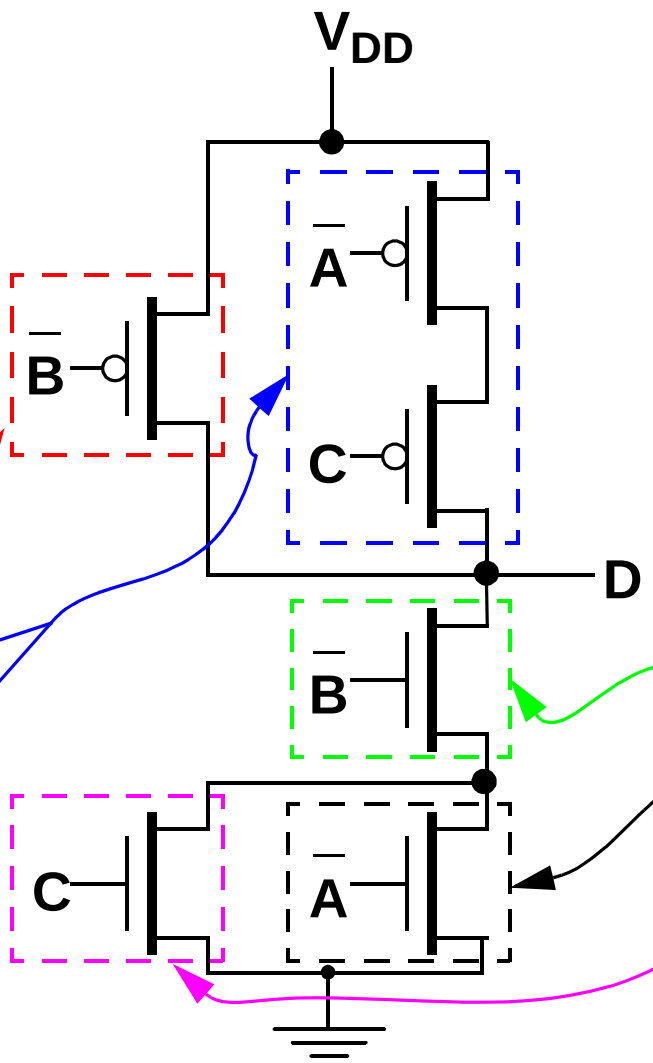
- Can this be implemented without the extra inverter at the output? Answer: Yes!

SWITCH NETWORKS

PULL-UP/PULL-DOWN

- SWITCH NETWORKS
  - -OR NETWORK
  - -XOR NETWORK
  - -XNOR NETWORK

$$D = A\bar{C} + B$$



PULL-UP

A	B	C	D
0	0	0	Z
0	0	1	Z
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	Z
1	1	0	1
1	1	1	1

PULL-DOWN

A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	Z
0	1	1	Z
1	0	0	Z
1	0	1	0
1	1	0	Z
1	1	1	Z

SWITCH NETWORKS

FUNCTION IMPLEMENTATION

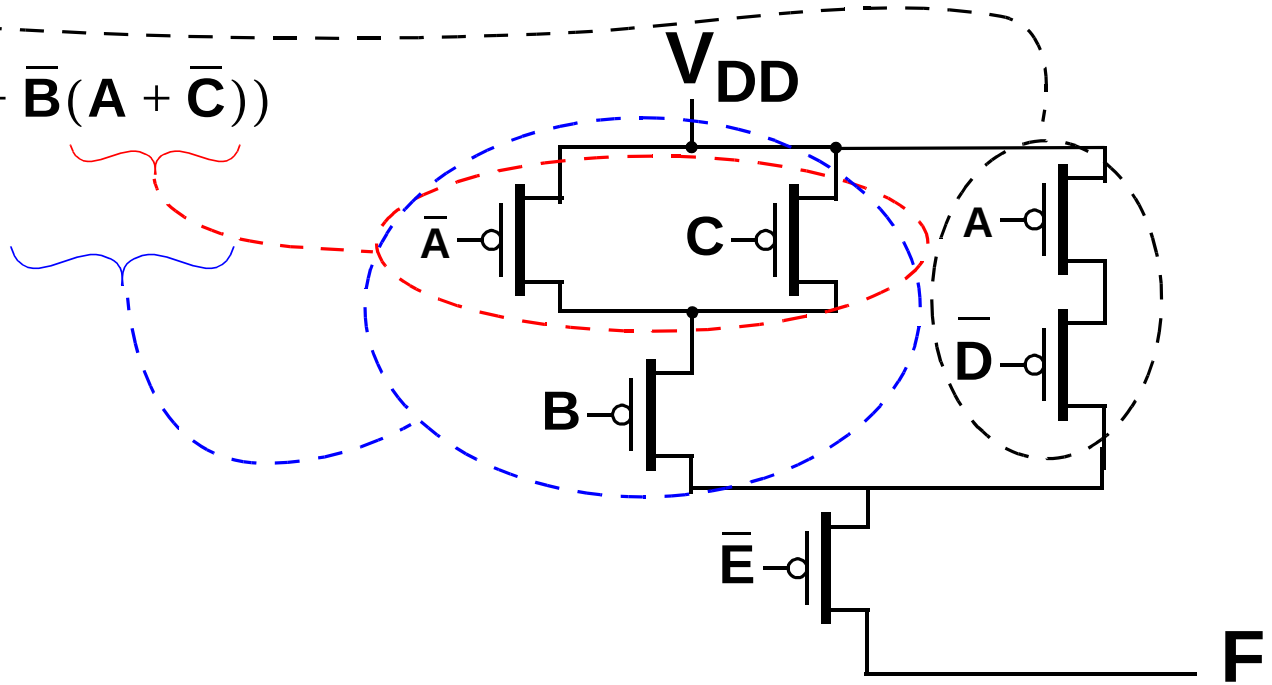
- Most Boolean functions can be easily implemented using switches.
- The basic rules are as follows
  - **Pull-up** section of switch network
    - **Use complements** for all literals in expression
    - Use only **pMOS devices**
    - Form **series** network for an **AND** operation
    - Form **parallel** network for an **OR** operation
  - **Pull-down** section of switch network
    - **Use complements** for all literals in expression
    - Use only **nMOS devices**
    - Form **parallel** network for an **AND** operation
    - Form **series** network for an **OR** operation

SWITCH NETWORKS

EXAMPLE PULL-UP

- To implement the Boolean function given below, the following pull-up network could be designed.

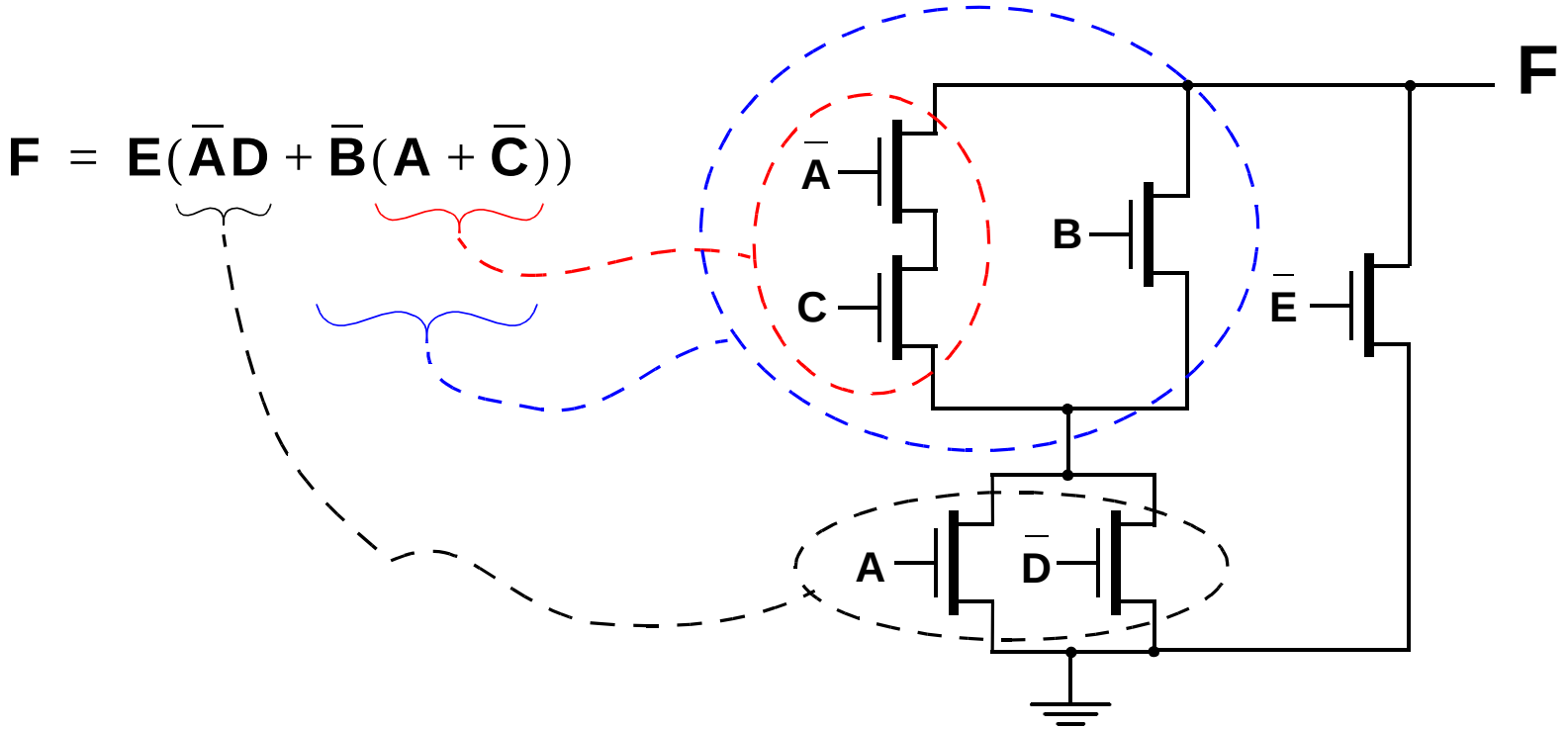
$$F = E(\overline{A}D + \overline{B}(A + \overline{C}))$$



SWITCH NETWORKS

EXAMPLE PULL-DOWN

- To complete the switch design, the pull-down section for the Boolean function must also be designed.



- Notice how **AND** and **OR** become **OR** and **AND** circuits, respectively.

SWITCH NETWORKS

COMPLETED EXAMPLE

- Putting the pull-up and pull-down pieces together gives the following CMOS switch implementation of the Boolean function.