Lecture -9
CMOS Logic, Digital System Design

Prepared By: Dr. Shahriyar Masud Rizvi

CMOS Logic

- The most widely used logic style is static complementary CMOS [1].
- The static CMOS style is really an extension of the static CMOS inverter to multiple inputs.

- The primary advantage of the CMOS structure is robustness (i.e., low sensitivity to noise), good performance, and low power consumption with no static power dissipation [1].

- The complementary CMOS circuit style falls under a broad class of logic circuits called static circuits in which at every point in time (except during the switching transients), each gate output is connected to either VDD or Vss via a low-resistance path [1].

- CMOS allows a much higher density of logic functions in a single chip compared to TTL.
- TTL circuits consume more power in comparison to CMOS circuits at rest.

CMOS Logic-Architecture

- A CMOS network is a combination of pull-up network (PUN) and pull-down network (PDN) [1].

- The function of the PUN is to provide a connection between V_{DD} and output when the function needs to be 1 [1].
- So, PUN is responsible for producing HIGH output.

- Similarly, the task of the PDN is to provide a connection between output and ground when function needs to be 0 [1].
- So, PDN is responsible for producing LOW output.

- The PUN and PDN are constructed in a mutually exclusive fashion such that one and only one of the networks is conducting in any state [1].

CMOS Logic-Reasoning behind choice of MOS for PUN & PDN

- Consider a logic device with a supply voltage $V_{DD} = 5V$, ground voltage $V_{SS} = 0V$ and inversion point (transition between logic-0 and logic-1) at $V_{DD}/2 = 2.5V$ and has an output named Y.
- So, Y = logic-0 when $0 \leq Y \leq 2.4$ and Y = logic-1 when $2.6 \leq Y \leq 5$.

- When Y is, say, 5V, it will be considered a strong logic-1. But when Y is, say, 4V, it will be considered a weak logic-1.
- Similarly, when Y is, say 0V, it will be considered a strong logic-0. But when Y is, say, 1V, it will be considered a weak logic-0.

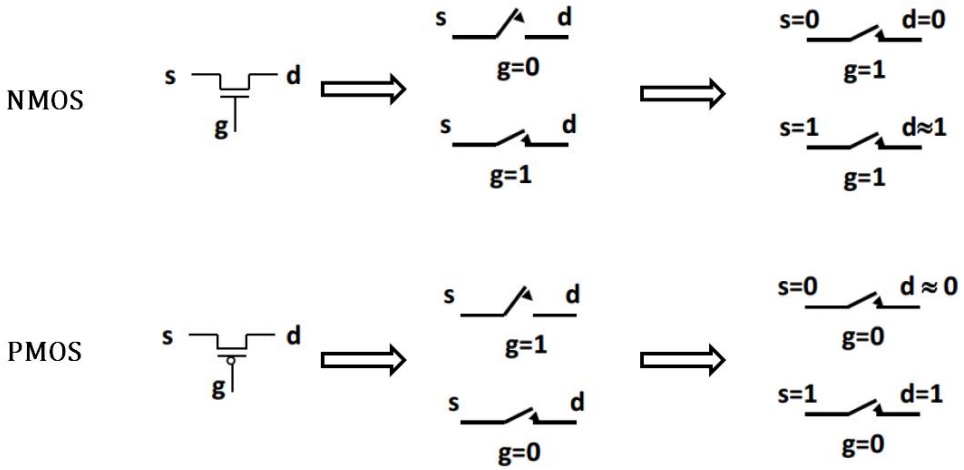
- An NMOS can provide a Y that is between V_{ss} and $V_{DD} - V_{THN}$, where V_{THN} is the NMOS threshold voltage. A PMOS can provide a Y that is between $|V_{THP}|$ and V_{DD}., where V_{THP} is the PMOS threshold voltage.
- Therefore, NMOS can produce strong logic-0 and weak logic-1.
- PMOS can produce strong logic-1 and weak logic-0.

CMOS Logic-MOS Logic-level Properties

- NMOS can produce strong logic-0 and weak logic-1.
- PMOS can produce strong logic-1 and weak logic-0.

- So, PUN (responsible for producing HIGH output state) is built from PMOS transistors and PDN (responsible for producing LOW output state) is built from NMOS transistors.

- Note that NMOS conducts when it receives a HIGH input (gate voltage (V_{GSN}) is HIGH).
- PMOS conducts when it receives a LOW input (gate voltage (V_{GSP}) is LOW).



CMOS Logic-Construction Rules

- In summary,
- CMOS logic realizes complimentary logic naturally. One can construct NOT, NAND, NOR in a single stage (need to add NOT to NAND, NOR to get AND, OR functions, respectively).
  In other words, the logic expression should have a bar (not) on the right-hand-side.

- CMOS logic is constructed from a PUN, consisting of PMOS transistors, that can produce HIGH output when needed and a PDN, consisting of NMOS transistors, that can produce LOW output when needed.
- So, either PUN or PDN is active at one time.
- Mathematically, one can express the output equation the following way [2].

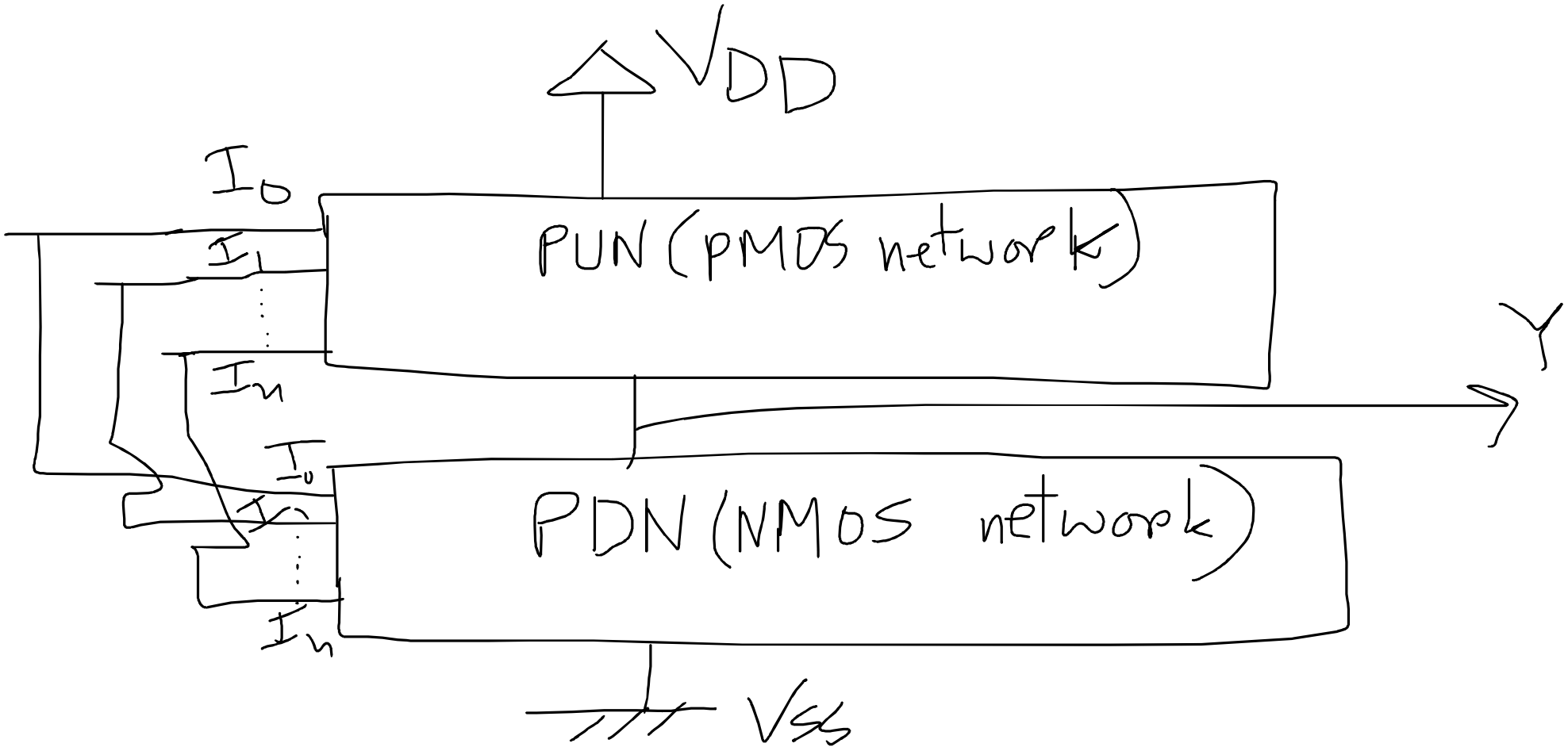
$$Y = f(\overline{I_0}, \overline{I_1}, \overline{I_2} \ldots\ldots \overline{I_n}).1 + f(I_0, I_1, I_2 \ldots\ldots I_n).0$$

- Each input is connected 1 PMOS and 1 NMOS.

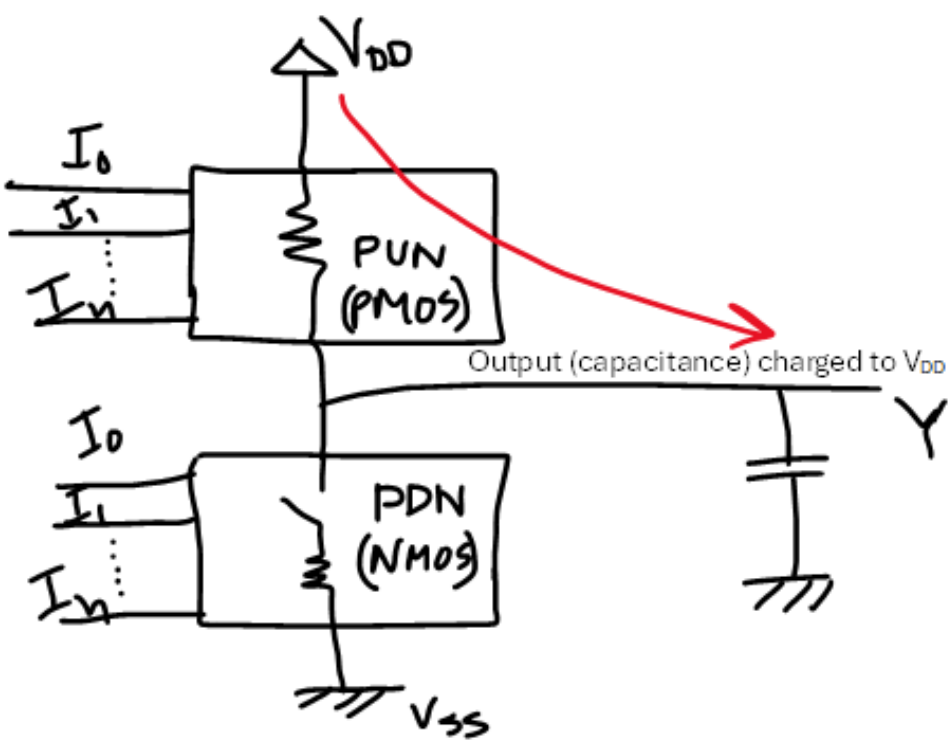
- So, number of transistors in a CMOS logic = 2N, where N = no. of inputs [1].

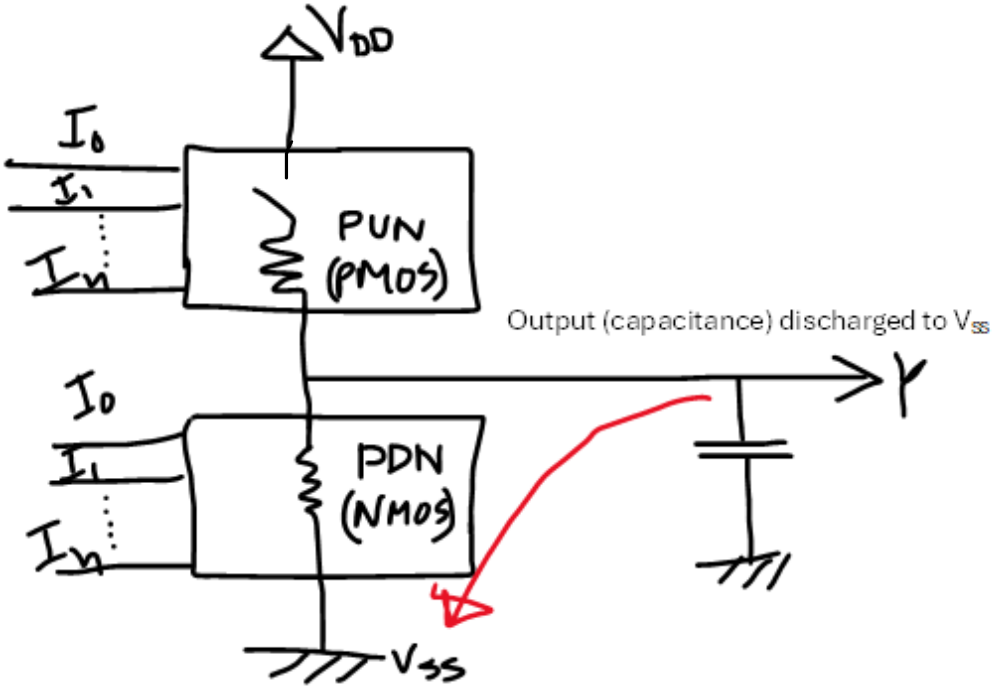
CMOS Logic-Generic Architecture

$$Y = f(\overline{I_0}, \overline{I_1}, \overline{I_2} \ldots\ldots \overline{I_n}).1 + f(I_0, I_1, I_2 \ldots\ldots I_n).0$$

- 

CMOS Logic-Pull-Up and Pull-Down Behavior



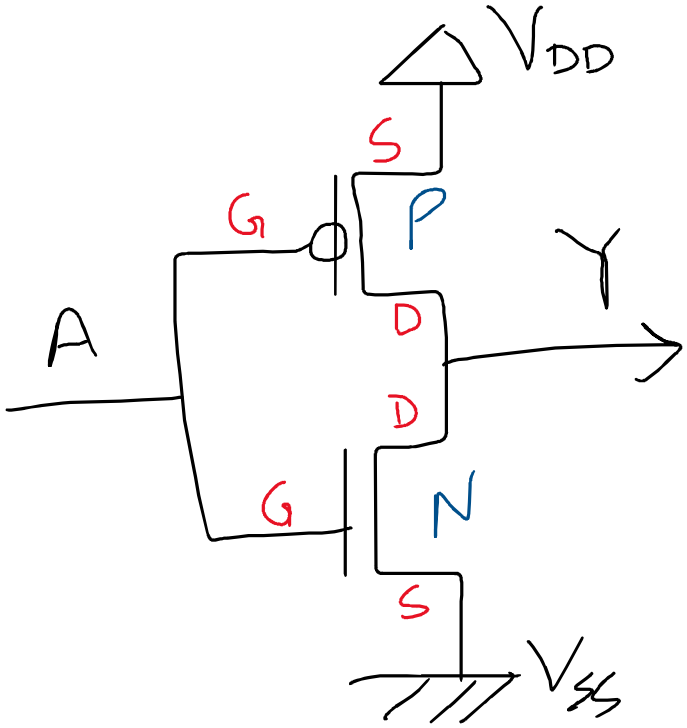


CMOS Logic-NOT Gate

- NOT gate

A	Y	NMOS (N)	PMOS (P)	Remarks
0	1 (V_{DD})	OFF	ON	Output (capacitance) charged to supply voltage (V_{DD})
1	0 (V_{SS})	ON	OFF	Output (capacitance) discharged to ground voltage (V_{SS})

- For PDN, identify inputs that produces 0. $\bar{Y} = A$,
- So, PDN has just 1 NMOS connected to A.
- For PUN, identify inputs that produces 1. $Y = \bar{A}$,
- So, PUN has just 1 PMOS connected to A.
- Note that since PMOS activates for LOW input, one needs to connect A, rather than $\bar{A}$.



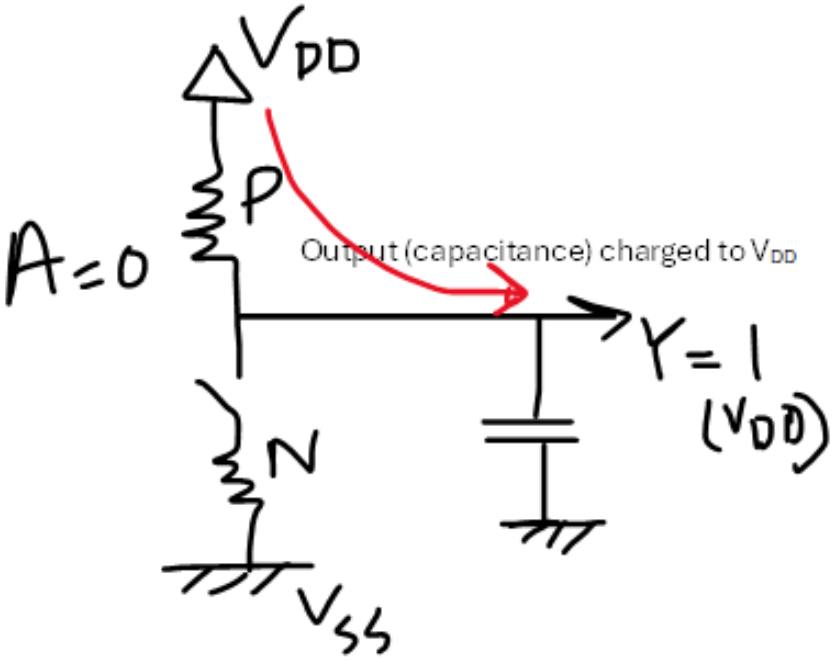
Note that we need NMOS to produce LOW output.
NMOS conducts for HIGH input. So, PDN must be built from NMOS.

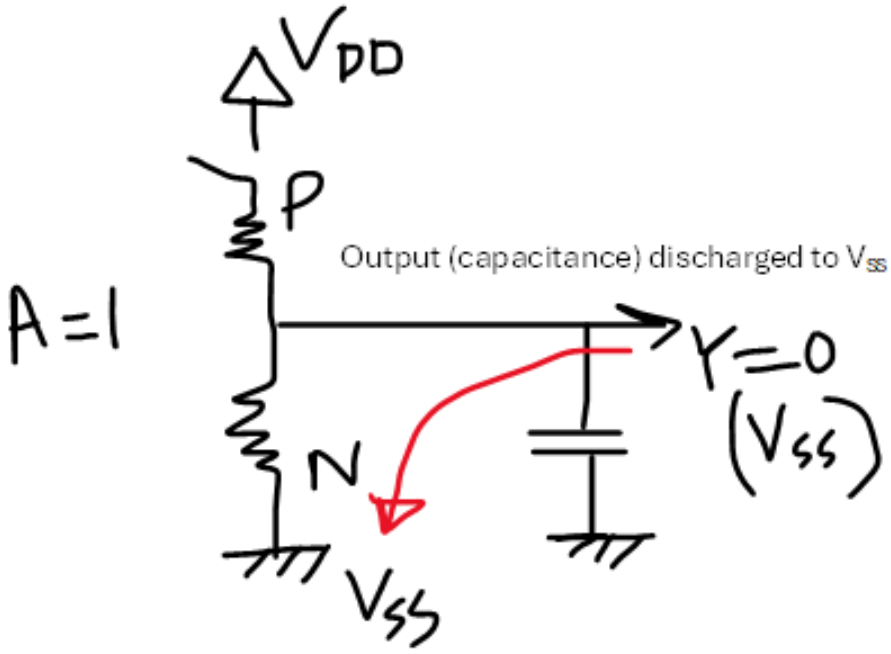
Similarly, we need PMOS to produce HIGH output.
PMOS conducts for LOW input. So, PUN must be built from PMOS.

Finally, $Y = A.0 + 1.\bar{A}$.

CMOS Logic-NOT Gate-Conduction in PUN and PDN

- NOT gate

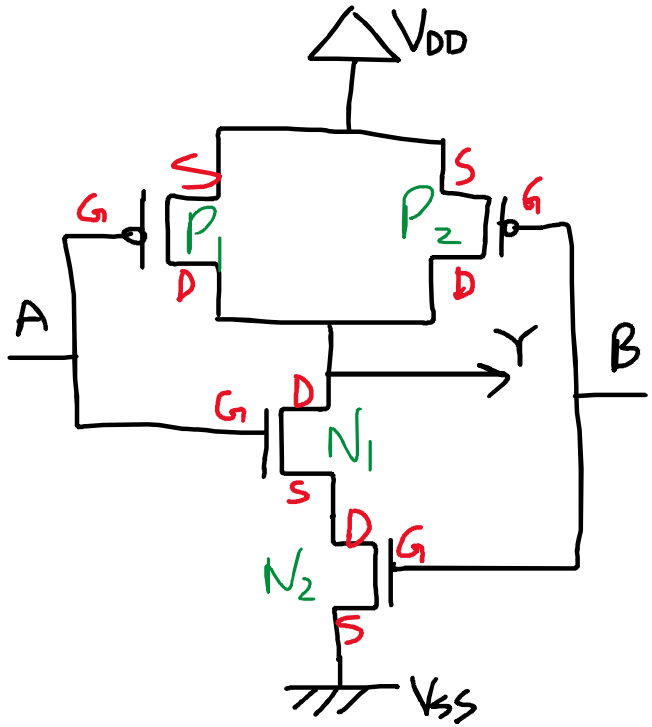




CMOS Logic-NAND Gate

- NAND gate

A	B	Y	N1	N2	P1	P2
0	0	1	OFF	OFF	ON	ON
0	1	1	OFF	ON	ON	OFF
1	0	1	ON	OFF	OFF	ON
1	1	0	ON	ON	OFF	OFF



- Mathematical Approach:
- For PDN, $\bar{Y} = A.B$, So, $\boldsymbol{Y = \overline{A.B}}$.
- So, A and B should be connected in series for PDN.

- For PUN, $Y = \bar{A}.\bar{B} + \bar{A}.B + A.\bar{B} \Rightarrow \boldsymbol{Y = \bar{A} + \bar{B}}$
  - One can also get Y for PUN by applying DeMorgan's duality to Y obtained from PDN. $\boldsymbol{Y = \overline{A.B} = \bar{A} + \bar{B}}$.

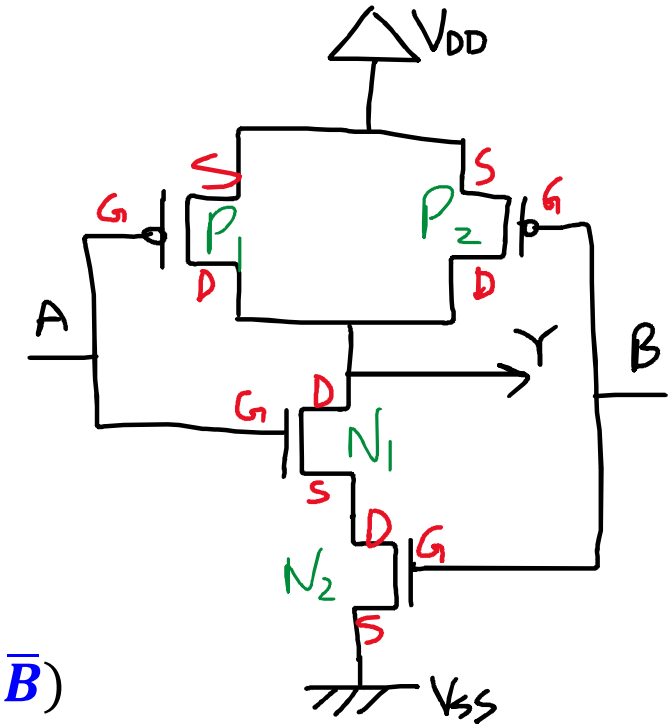
So, A and B should be connected in parallel for PUN.

Note that since PMOS transistors activate on LOW input, we need to provide A, B, to PUN rather than not $\bar{A}$, $\bar{B}$.

CMOS Logic-NAND Gate

- NAND gate

A	B	Y	N1	N2	P1	P2
0	0	1	OFF	OFF	ON	ON
0	1	1	OFF	ON	ON	OFF
1	0	1	ON	OFF	OFF	ON
1	1	0	ON	ON	OFF	OFF



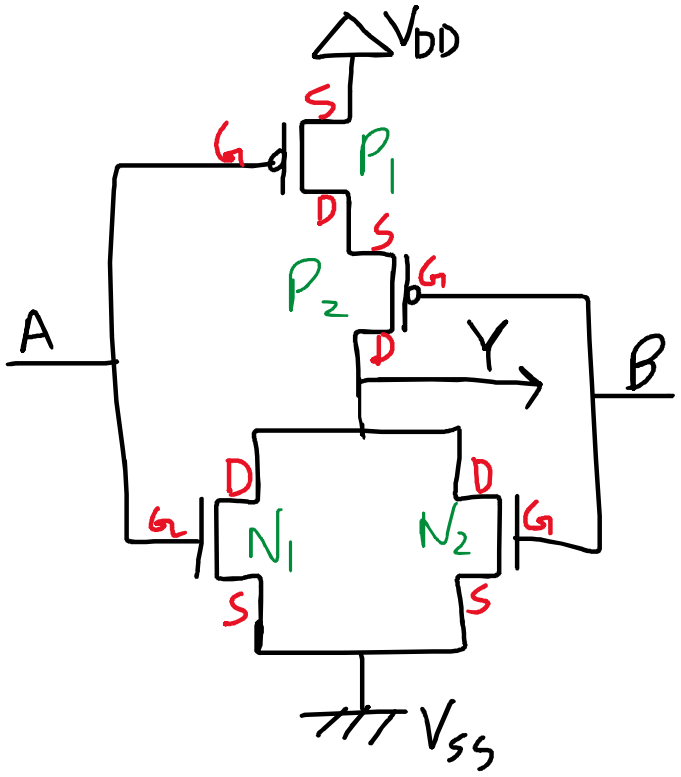
- Observational Approach:
- Note that Y is HIGH, when any one of the inputs is LOW.
- So, PMOS devices, who provide HIGH output, should be connected in parallel. (This happens since $\boldsymbol{Y = \overline{A} + \overline{B}}$)

- Y is LOW, when all the inputs are HIGH.
- So, NMOS devices, who provide LOW output, should be connected in series. (This happens since $\boldsymbol{Y = \overline{A.B}}$)

CMOS Logic-NOR Gate

- NOR gate

A	B	Y	N1	N2	P1	P2
0	0	1	OFF	OFF	ON	ON
0	1	0	OFF	ON	ON	OFF
1	0	0	ON	OFF	OFF	ON
1	1	0	ON	ON	OFF	OFF



- Mathematical Approach:
- For PDN, $\bar{Y} = \bar{A}.B + A.\bar{B} + A.B = A + B$, So, $\boldsymbol{Y = \overline{A + B}}$.
- So, A and B should be connected in parallel for PDN.

- For PUN, $\boldsymbol{Y = \bar{A}.\bar{B}}$
  - One can also, get Y for PUN by applying DeMorgan's duality to Y obtained from PDN. $\boldsymbol{Y = \overline{A + B} = \bar{A}.\bar{B}}$.

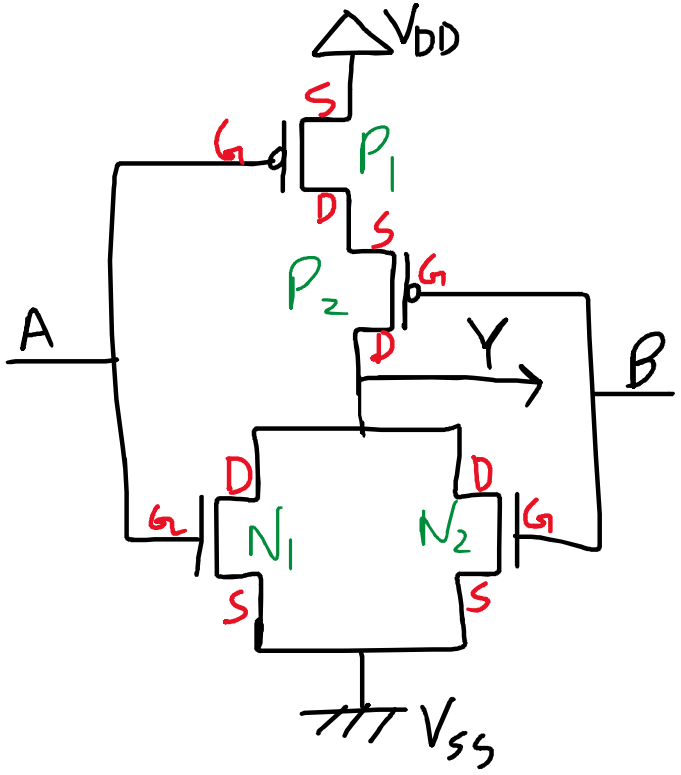
So, A and B should be connected in series for PUN.

Note that since PMOS transistors activate on LOW input, we need to provide A, B, to PUN rather than not $\bar{A}$, $\bar{B}$.

CMOS Logic-NOR Gate

- NOR gate

A	B	Y	N1	N2	P1	P2
0	0	1	OFF	OFF	ON	ON
0	1	0	OFF	ON	ON	OFF
1	0	0	ON	OFF	OFF	ON
1	1	0	ON	ON	OFF	OFF



- Observational Approach:
- Note that Y is HIGH, when all the inputs are LOW.
- So, PMOS devices, who provide HIGH output, should be connected in series. (This happens since $Y = \overline{A}.\overline{B}$)

- Y is LOW, when any one of the inputs is HIGH.
- So, NMOS devices, who provide LOW output, should be connected in parallel. (This happens since $Y = \overline{A + B}$)

CMOS Logic-Implementing Logic Functions

- **To implement a logic function with CMOS logic, one needs to follow the general rules of CMOS logic as mentioned below.**

- 1. CMOS logic realizes complimentary logic naturally.
  In other words, the logic expression should have a bar (not) on the right-hand-side.
- 2. CMOS logic is constructed from a PUN, consisting of PMOS transistors, that can produce HIGH output when needed and a PDN, consisting of NMOS transistors, that can produce LOW output when needed.
  Mathematically, one can express the output equation the following way [2].

$$Y = f(\overline{I_0}, \overline{I_1}, \overline{I_2} \ldots\ldots \overline{I_n}).1 + f(I_0, I_1, I_2 \ldots\ldots I_n).0$$

- 3. Each input is connected 1 PMOS and 1 NMOS.
  - So, number of transistors in a CMOS logic = 2N, where N = no. of inputs [1].

- **In addition, one needs to follow the following rule (this rule is a direct consequence of PDN and PUN being dual networks (related by DeMorgan's duality)).**
- 4a. For PDN, AND function is realized with series connection.
  OR function is realized with parallel connection.
- 4b. For PUN, AND function is realized with parallel connection.
  OR function is realized with series connection.

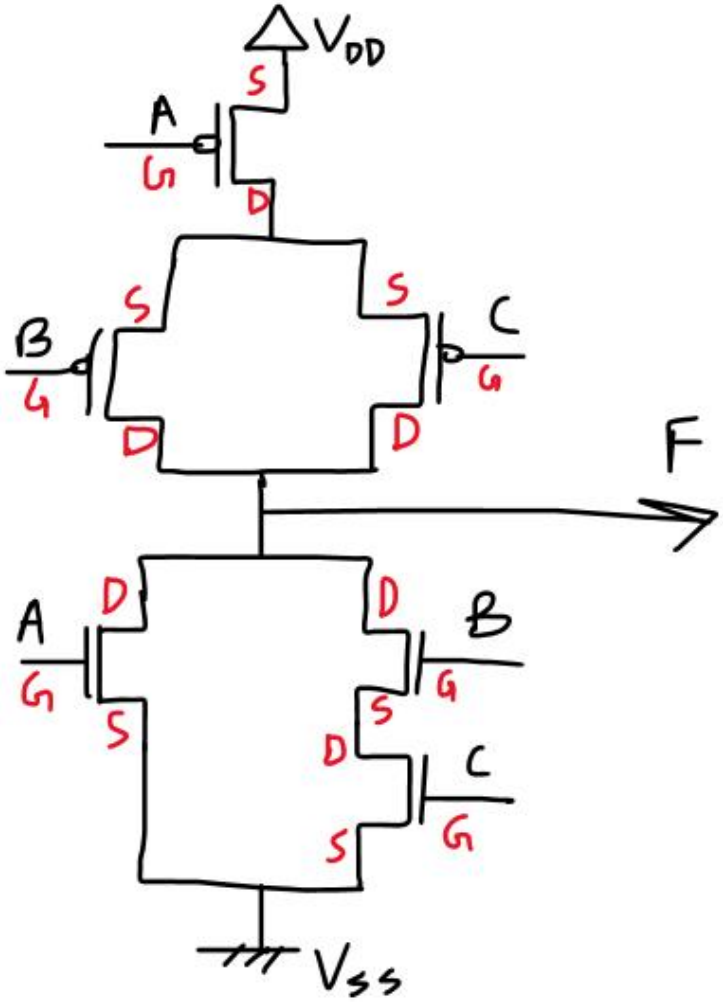
CMOS Logic-Implementing Logic Functions

- Implement $F = \overline{A + B.C}$ in CMOS logic.

- Reminder about Rule 4:
- 4a. For PDN, AND function is realized with series connection.
  OR function is realized with parallel connection.
- 4b. For PUN, AND function is realized with parallel connection.
  OR function is realized with series connection.

- Note that Rule 4 is a direct consequence of DeMorgan's
- Duality Theorem.

- PDN, $F = \overline{A + B.C}$
- PUN, $F = \overline{A + B.C} = \overline{A} \,.\, (\overline{B.C}) \Rightarrow F = \overline{A} \,.\, (\overline{B} + \overline{C})$



CMOS Logic-Closing Remarks

- Complimentary CMOS logic is a naturally inverting logic. That is, one can construct NOT, NAND and NOR gates in a single stage.
- If one needs to build a buffer, AND and OR gates, one needs to connect an inverter to the output of NOT, NAND and NOR, respectively. In other words, 2-stage logic is required here.

- One can see that a NAND gate requires 4 transistors in CMOS logic.
- An AND gate will require 6 transistors (4 for NAND and 2 for NOT).

- In CMOS logic, it's beneficial to express logic functions in terms of NAND and NOR (or have bar on the right-hand-side expression), rather than AND and OR as it reduces number of transistors and hence area consumed by the design.

- DeMorgan's Duality Theorem allows us to represent logic functions in terms of NAND and NOR as well as develop dual equations of PDN and PUN that allow direct implementation of logic functions in CMOS logic.

-

Digital System Design

Designing a digital system requires following the steps to be performed:

- Look for the problem statement.
- Find out the inputs and outputs of the system.
- Related the input of your system with the outputs.
- Develop a truth table/ behavior table for your system using input and output relationship that you have established.
- From your truth-table generate a standard output expression, relating the inputs to the outputs.
- Reduce the output expression using either Boolean Algebra or K-MAP.
- Write down your reduced expression (minimized expression)
- Design the system circuit with combinational logic gates.
- Convert the combinational logic gates to transistor level schematic (CMOS schematic)

References

1. Jan Rabaey, Digital Integrated Circuits—A Design Perspective, 2nd Edition, Pearson.
2. John Uyemura, CMOS Logic Circuit Design, Kluwer Academic Publishers.
3. Thomas L. Floyd, “Digital Fundamentals” 11th edition, Prentice Hall – Pearson Education.

Thank You