Digital CMOS Logic Design

Somesh Kr Malhotra

Assistant Professor

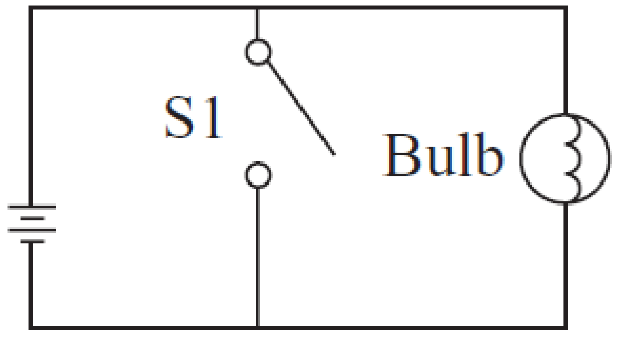
ECE Department,UIET

Introduction

- We introduce the digital logic circuit design using CMOS transistors,which are most popular because of low power dissipation and less area requirement compared to any other logic circuits.
- It is worthwhile mentioning that almost 90% of the total semiconductor devices are fabricated using silicon CMOS technology.
- CMOS logic is a combination of nMOS and pMOS logic. The nMOS and pMOS transistors are both functionally and structurally complement to each other.

Digital Logic Design

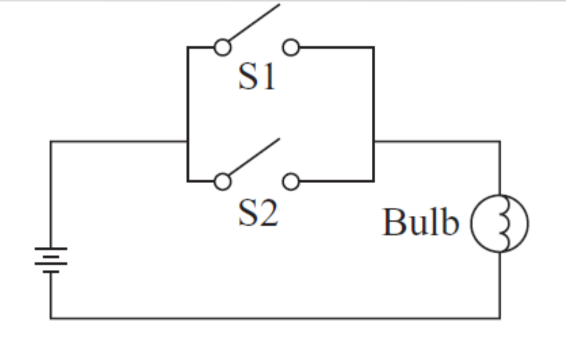
- In digital logic design, there are three primary logic operations: (a) NOT, (b) OR,and (c) AND. Using these three primary logics, any other logic such as NAND,NOR, XOR, or XNOR can be derived.
- In NOT logic, if the input (*A*) is TRUE, the output (*F*) is FALSE; and if the input (*A*) is FALSE, the output (*F*) is TRUE. The NOT logic can be realized using a simple circuit as shown in Fig. 1(a). When the switch S1 is ON, the bulb will not glow. When the switch S1 is OFF, the bulb will glow.



(a)

Digital Logic Design

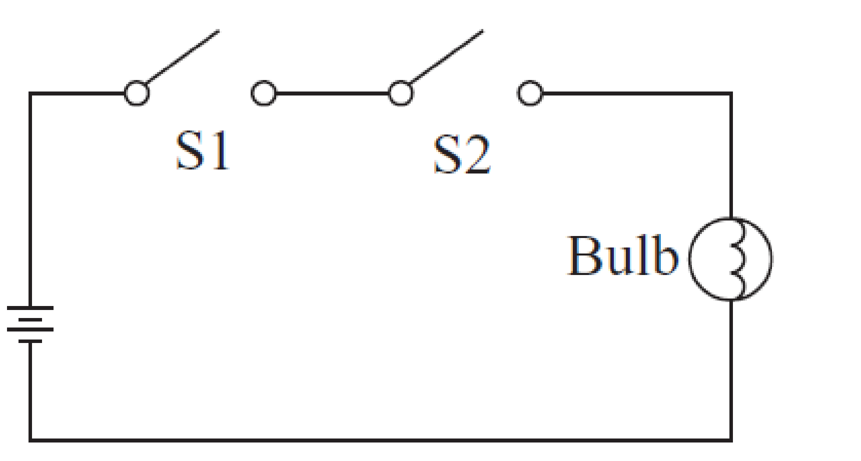
- In OR logic, if both the inputs are FALSE, the output is FALSE. Otherwise, the output is TRUE. The OR logic can be realized using two switches S1 and S2 connected in parallel, as shown in Fig. 1(b).
- When both switches are OFF, the bulb will not glow; otherwise, the bulb will glow.



(b)

Digital Logic Design

- In AND logic, if both the inputs are TRUE, the output is TRUE. Otherwise, the output is FALSE. The AND logic can be realized using two switches S1 and S2 connected in series as shown in Fig. 1(c).

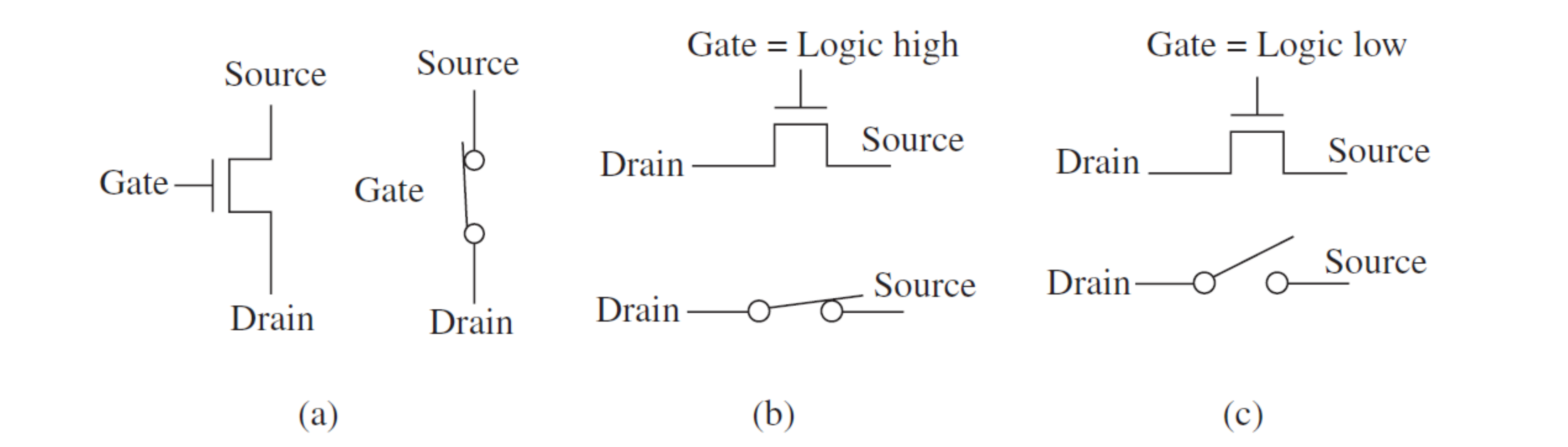


(c)

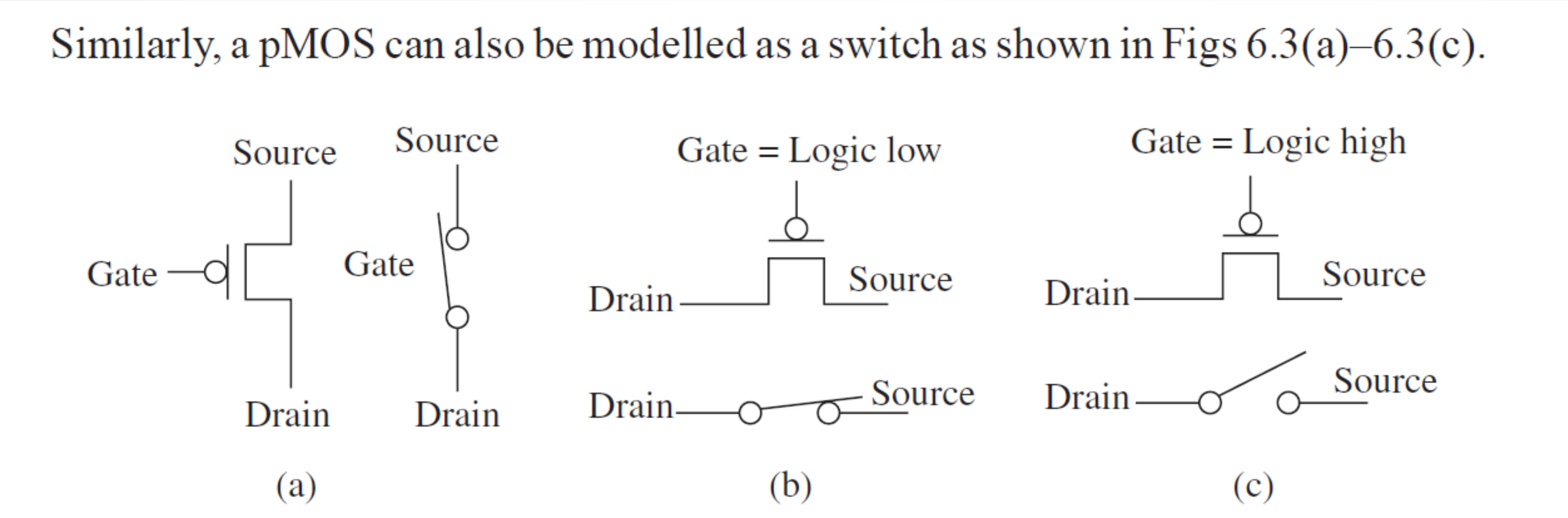
Digital Logic Design

- The mechanical switches shown in Fig.1 can be replaced by MOS transistors as the MOS transistor behaves as a switch when it is operated between the cut-off and saturation regions.
- As shown in Fig. 2(a), an nMOS transistor can be modelled as a switch connected between the drain (D) and source (S) and the switch is controlled by gate (G).
- When the gate is logic high (H), the nMOS is ON and the switch is closed, and D is connected to S [see Fig. 2(b)].
- When the gate is logic low (L), the nMOS is OFF and the switch is open, and D is disconnected from S [see Fig. 2(c)].

Digital Logic Design

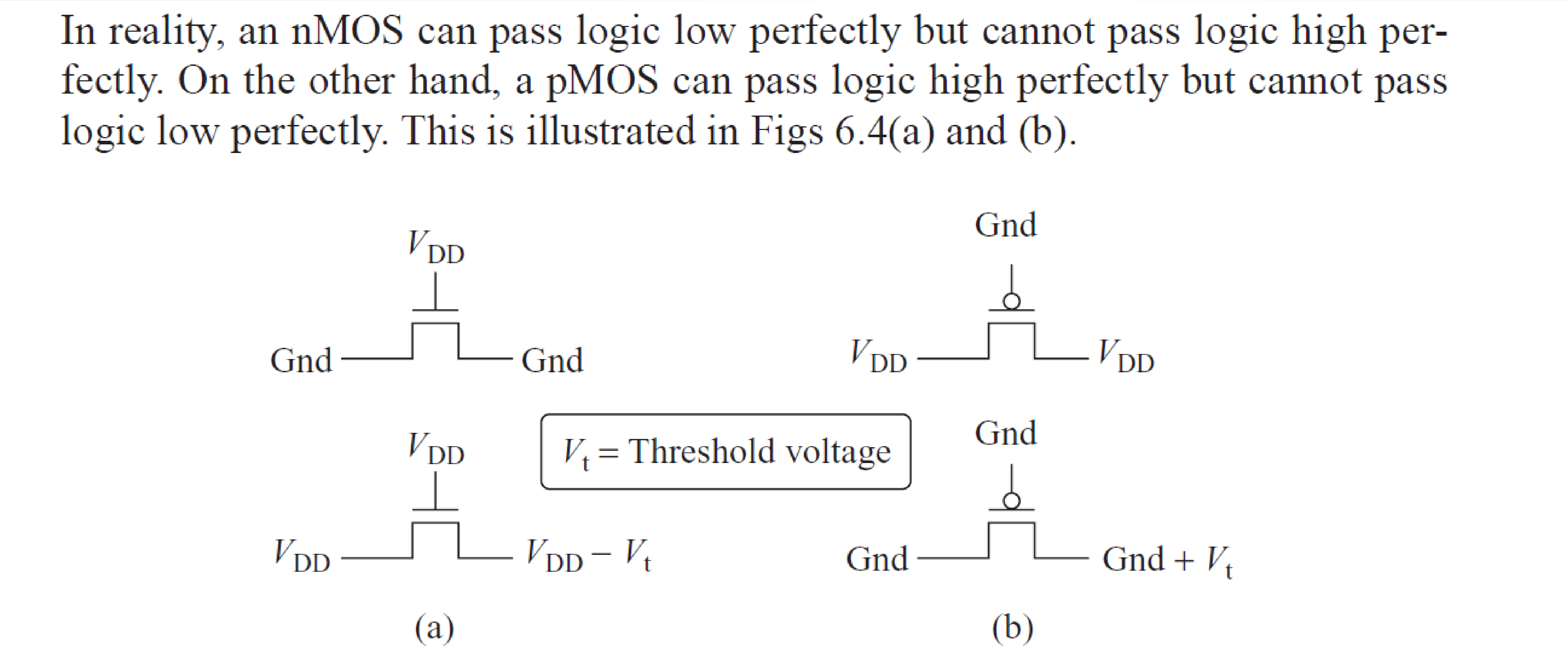


Digital Logic Design



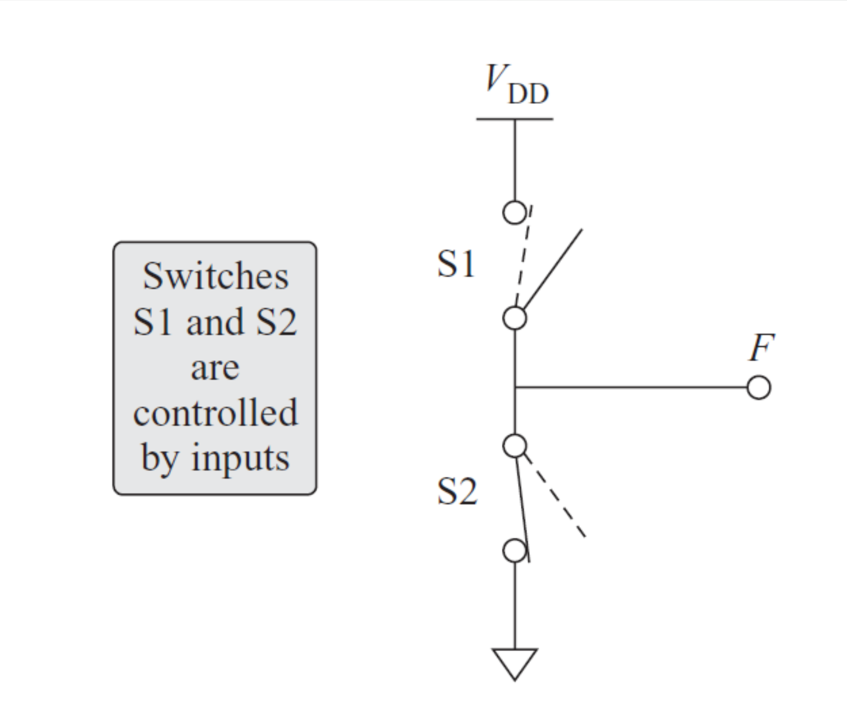
Similarly, a pMOS can also be modelled as a switch as shown in Figs 6.3(a)–6.3(c).

(a) (b) (c)



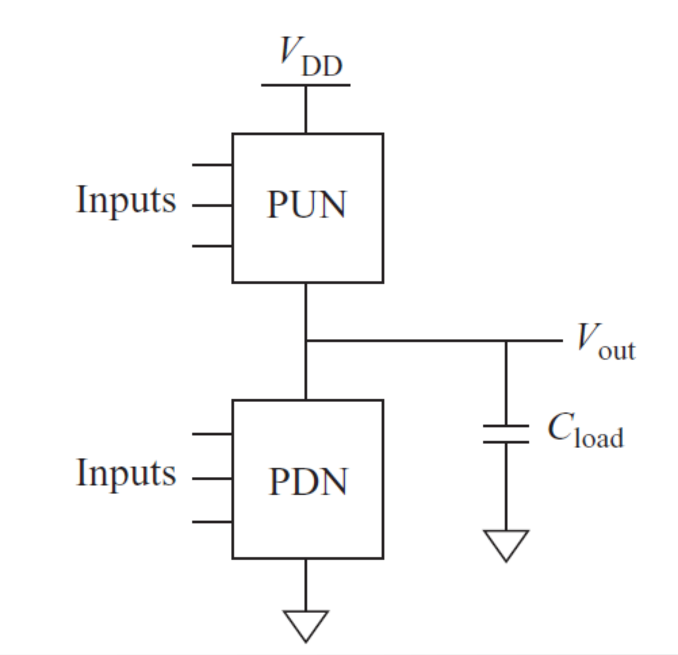
In reality, an nMOS can pass logic low perfectly but cannot pass logic high perfectly. On the other hand, a pMOS can pass logic high perfectly but cannot pass logic low perfectly. This is illustrated in Figs 6.4(a) and (b).

CMOS logic Design



- Any Boolean logic function (F) has possible values: either logic 0 or logic 1.
- For some of the input combinations, F = 1 and for all other input combinations, F =0.
- So in general, any Boolean logic function can be realized using a structure as shown in Fig.
- The switch S1 is closed and the switch S2 is open for input combinations that produce F = 1.
- The switch S1 is open and the switch S2 is closed for other input combinations that produce F = 0.

CMOS logic Design



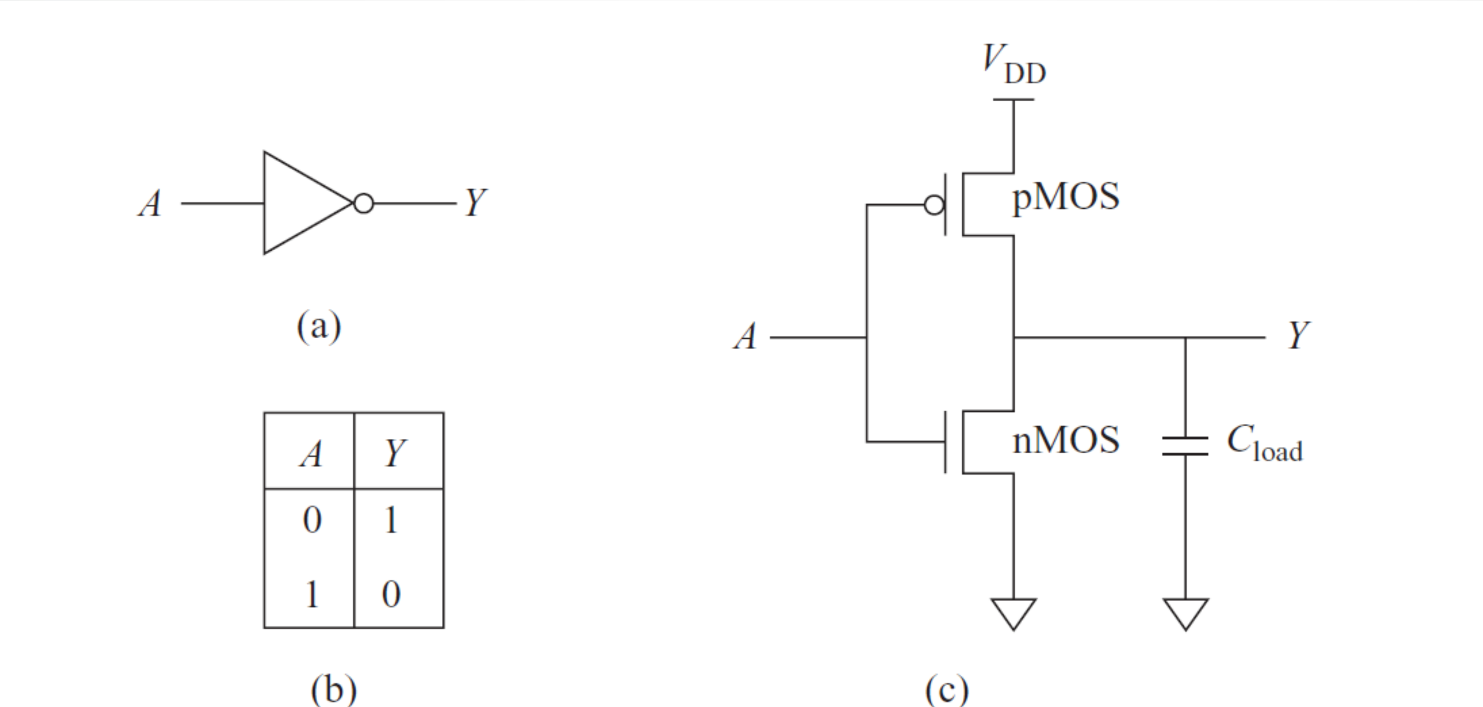
- As shown in Fig. the output (*F*) is either connected to *V*DD or to the ground, where the logic 0 is represented by the ground and the logic 1 is represented by *V*DD.
- So the basic requirement of digital logic design is to implement the pull-up switch (S1) and the pull-down switch (S2).
- As the pMOS transistors can pass logic 1 perfectly, they are used in pull-up switch realization.
- Similarly, as the nMOS transistors can pass logic 0 perfectly, they are used in pull-down switch realization.

CMOS design methodology

- The basic CMOS design methodology involves three steps:
    - Given the Boolean expression, take its complement
    - Design PDN by realizing
    - AND terms using series-connected nMOSFETs
    - OR terms using parallel-connected nMOSFETs
    - Design PUN just reverse (or dual) of the PDN

Design of CMOS Inverter (NOT) Gate

- A CMOS inverter is the simplest logic circuit that uses one nMOS and one pMOS transistor. The nMOS is used in PDN and the pMOS is used in the PUN, as shown in Figs (a)–(c).



(a)

A	Y
0	1
1	0

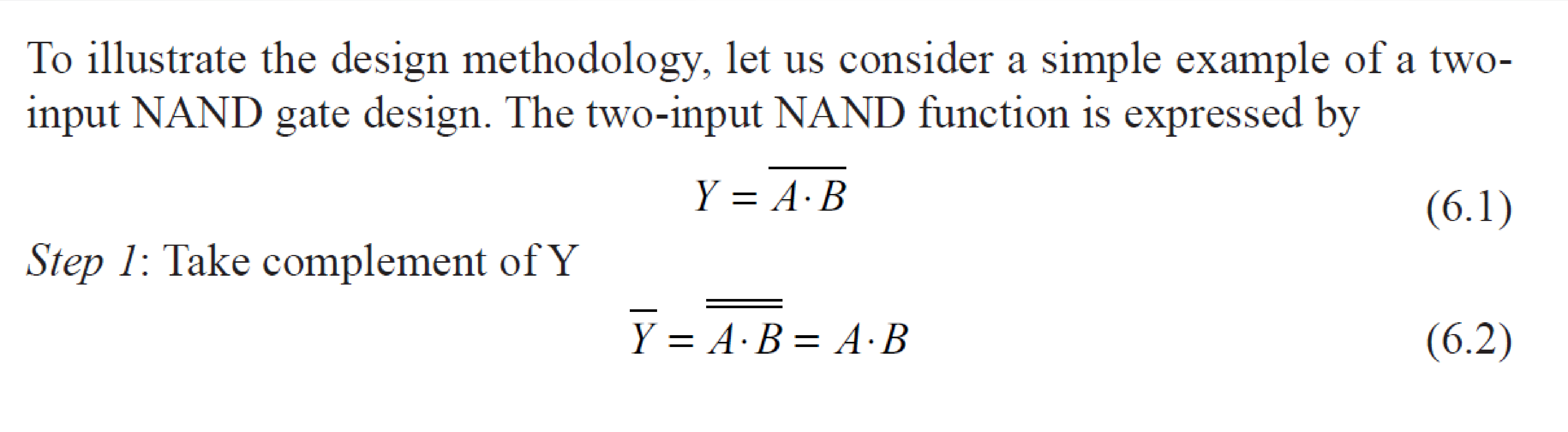
(b)

(c)

Design of CMOS Inverter (NOT) Gate

- **Operation** When input is low, the nMOS is OFF and the pMOS is ON. Hence,the output is connected to *V*DD through pMOS.
- When the input is high, the nMOS is ON and the pMOS is OFF. Hence, the output is connected to the ground through nMOS.
- We can connect a capacitor at the output node as shown in Fig. to represent the load seen by the inverter. The load capacitor is charged to *V*DD through pMOS when the input is low and is discharged to the ground through nMOS when the input is high.

Design of Two-input NAND Gate



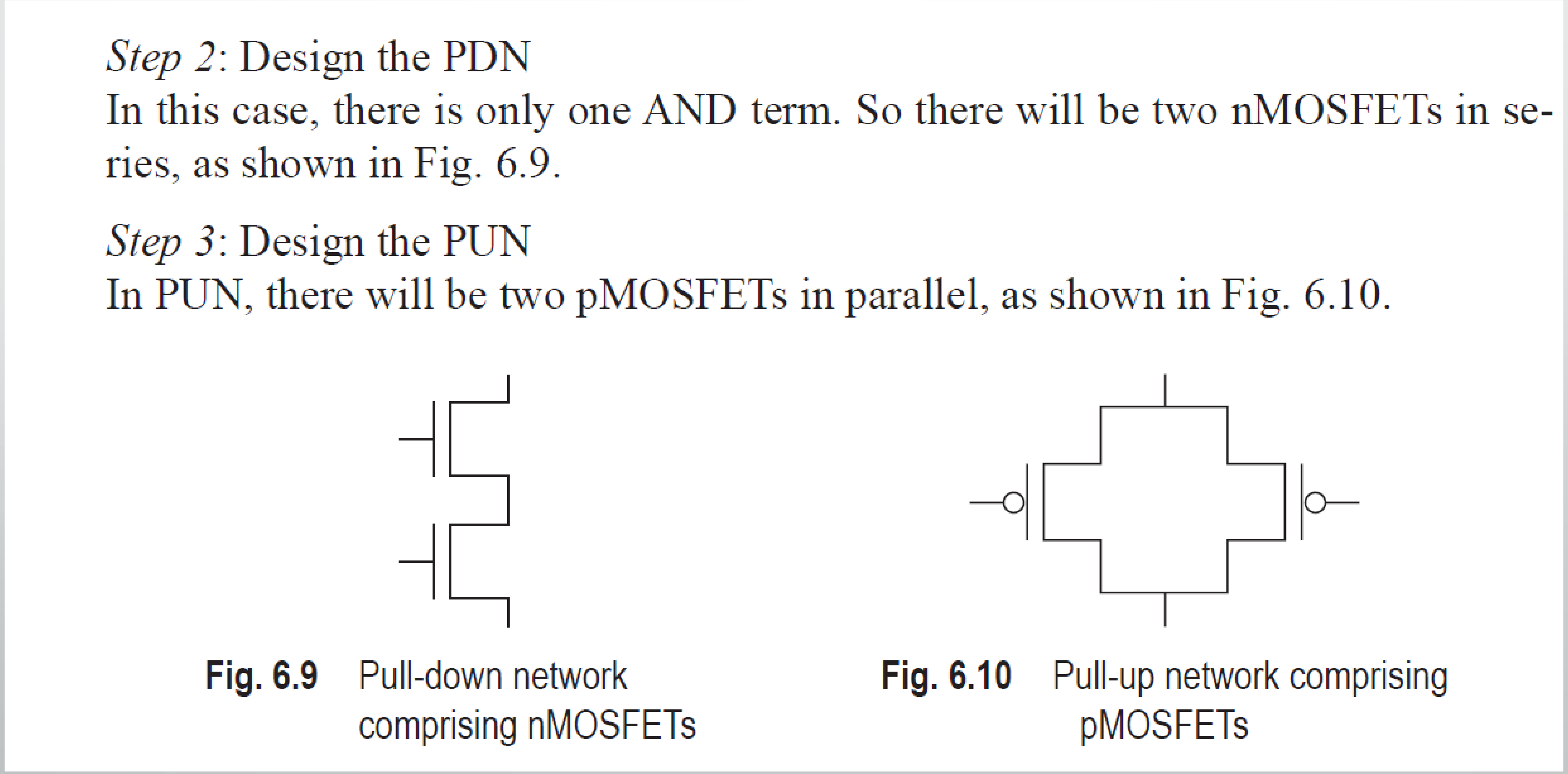
To illustrate the design methodology, let us consider a simple example of a two-input NAND gate design. The two-input NAND function is expressed by

$$Y = \overline{A \cdot B} \tag{6.1}$$

Step 1: Take complement of Y

$$\overline{Y} = \overline{\overline{A \cdot B}} = A \cdot B \tag{6.2}$$

Design of Two-input NAND Gate



Step 2: Design the PDN

In this case, there is only one AND term. So there will be two nMOSFETs in series, as shown in Fig. 6.9.

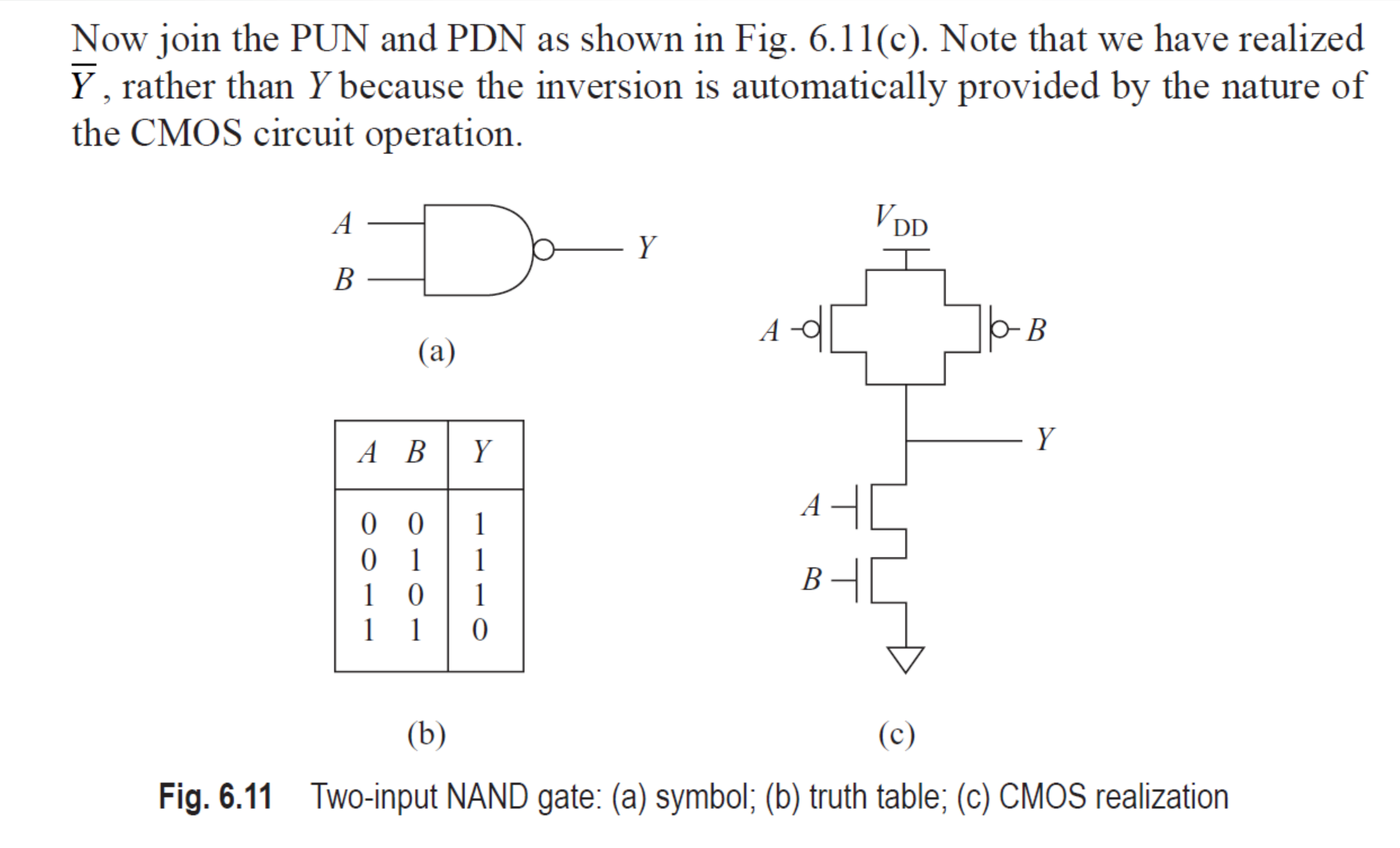
Step 3: Design the PUN

In PUN, there will be two pMOSFETs in parallel, as shown in Fig. 6.10.

Fig. 6.9 Pull-down network comprising nMOSFETs

Fig. 6.10 Pull-up network comprising pMOSFETs

Design of Two-input NAND Gate



Now join the PUN and PDN as shown in Fig. 6.11(c). Note that we have realized $\overline{Y}$, rather than Y because the inversion is automatically provided by the nature of the CMOS circuit operation.

A
B
Y
(a)
V_{DD}
A
B
Y
A
B
(c)

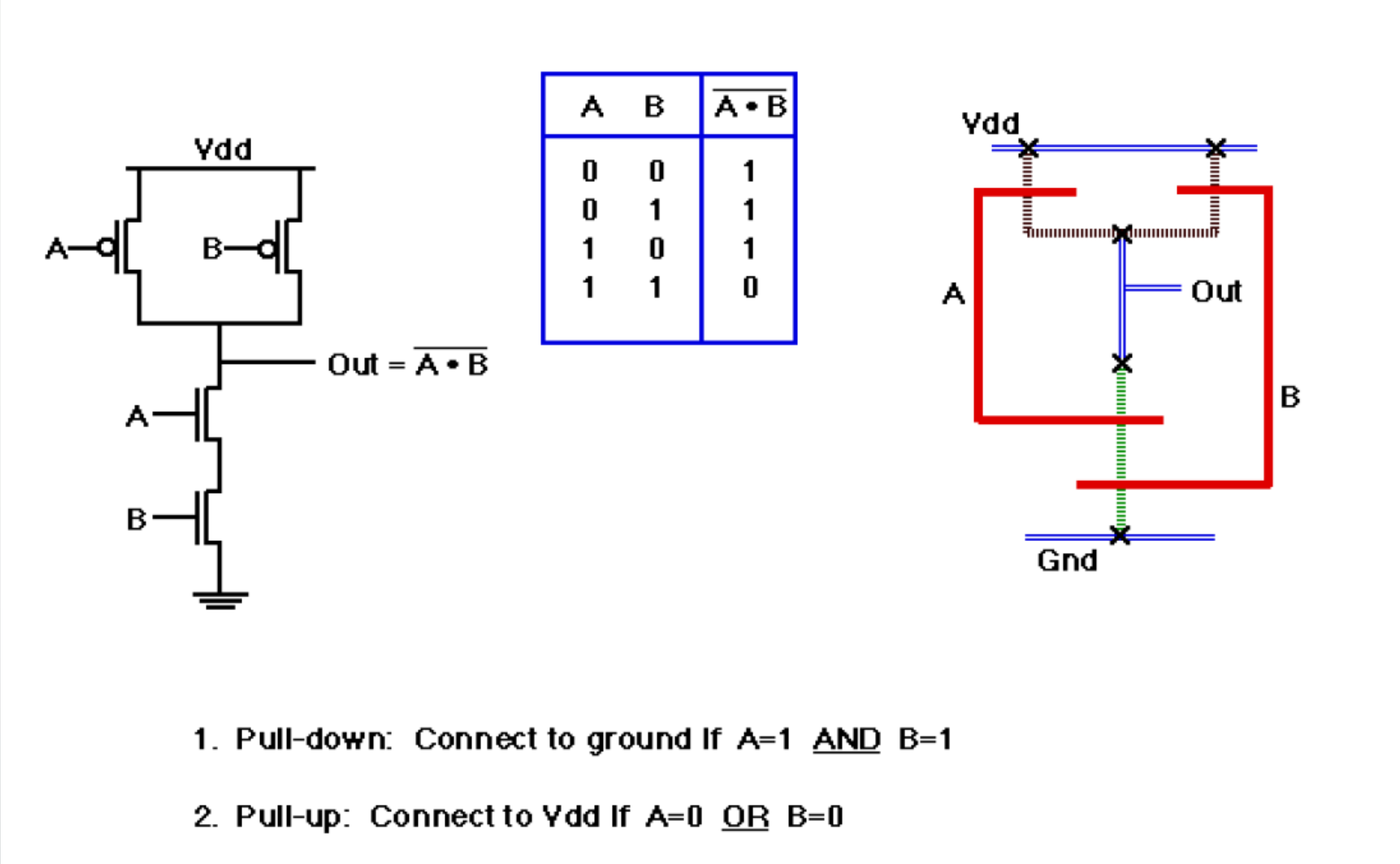
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

(b)

Fig. 6.11 Two-input NAND gate: (a) symbol; (b) truth table; (c) CMOS realization

Design of Two-input NAND Gate

- **Operation** When $A = 0$ and $B = 0$, both the nMOS transistors are OFF and both pMOS transistors are ON. Hence, the output is connected to VDD and we get logic high at the output.
- When $A = 1$ and $B = 0$, the upper nMOS is ON and lower nMOS is OFF. So, output cannot be connected to the ground. Under this condition, left pMOS is OFF but right pMOS is ON. Hence, the output is connected to VDD, and we get logic high at the output.
- When $A = 0$ and $B = 1$, the upper nMOS is OFF and lower nMOS is ON. So,output cannot be connected to ground. Under this condition, left pMOS is ON but right pMOS is OFF. Hence, the output is connected to VDD, and we get logic high at the output.
- When $A = 1$ and $B = 1$, both nMOS transistors are ON and both pMOS transistors are OFF. Hence, the output is connected to the ground, and we get logic low at the output. This is illustrated in Figs 6.11(a) and (c). This proves by the truth table of NAND gate shown in Fig. 6.11(b).

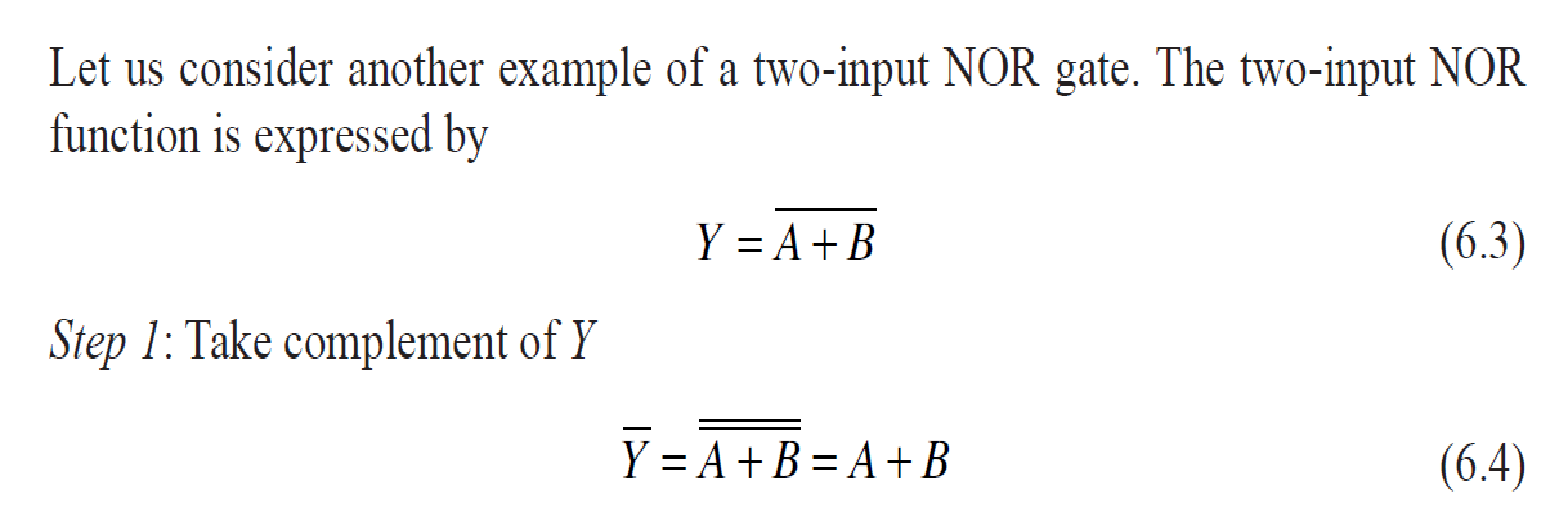


A	B	$\overline{A \bullet B}$
0	0	1
0	1	1
1	0	1
1	1	0

1. Pull-down: Connect to ground If A=1 AND B=1

2. Pull-up: Connect to Vdd If A=0 OR B=0

Design of Two-input NOR Gate



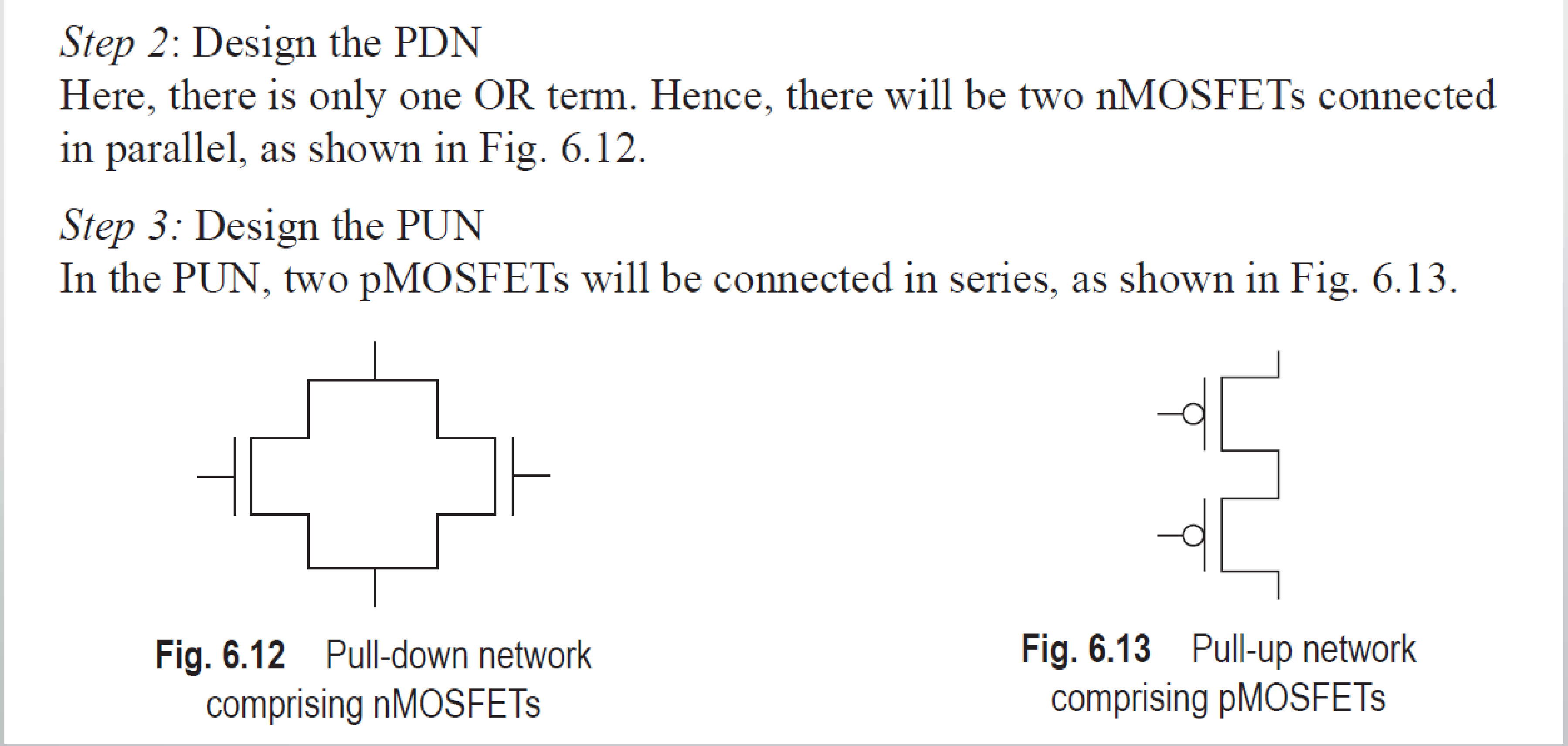
Let us consider another example of a two-input NOR gate. The two-input NOR function is expressed by

$$Y = \overline{A + B} \tag{6.3}$$

Step 1: Take complement of *Y*

$$\overline{Y} = \overline{\overline{A + B}} = A + B \tag{6.4}$$

Design of Two-input NOR Gate



Step 2: Design the PDN

Here, there is only one OR term. Hence, there will be two nMOSFETs connected in parallel, as shown in Fig. 6.12.

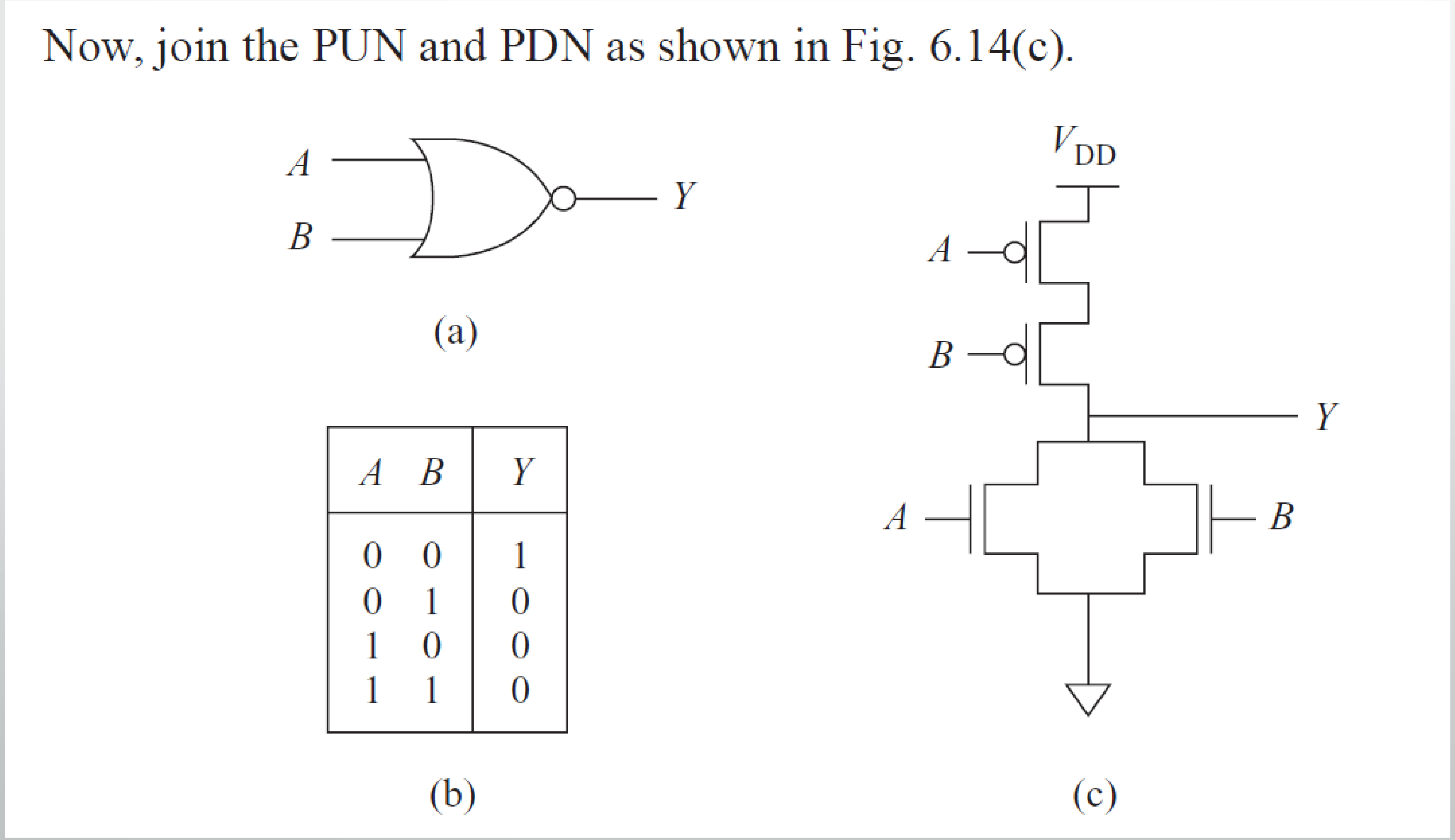
Step 3: Design the PUN

In the PUN, two pMOSFETs will be connected in series, as shown in Fig. 6.13.

Fig. 6.12 Pull-down network comprising nMOSFETs

Fig. 6.13 Pull-up network comprising pMOSFETs

Design of Two-input NOR Gate



Now, join the PUN and PDN as shown in Fig. 6.14(c).

(a)

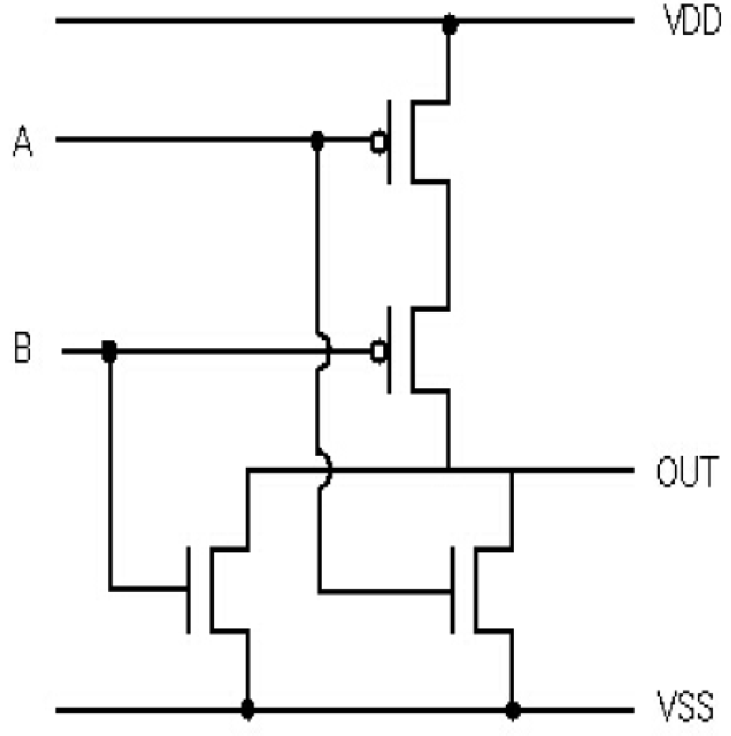
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

(b)

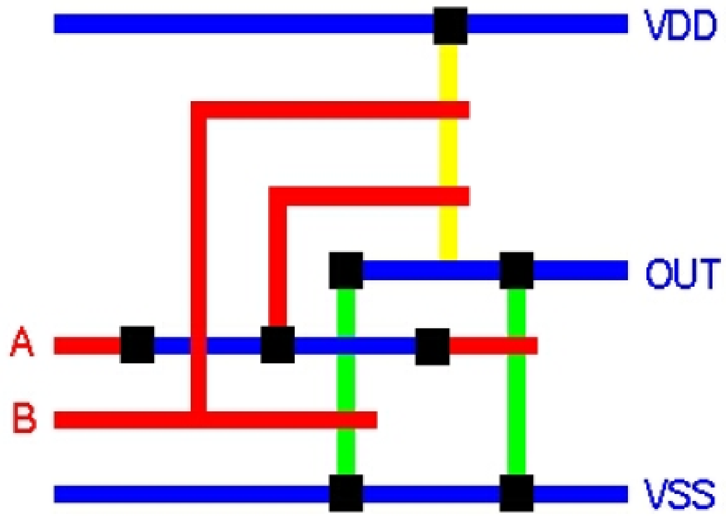
(c)

Design of Two-input NOR Gate

- **Operation** When *A* = 0 and *B* = 0, both nMOS transistors are OFF and both pMOS transistors are ON. Hence, the output is connected to *V*DD and we get logic high at the output.
- When *A* = 1 and *B* = 0, the upper pMOS is OFF and lower pMOS is ON. So, output cannot be connected to the *V*DD. Under this condition, left nMOS is ON and right nMOS is OFF. Hence, the output is connected to the ground and we get logic low at the output.
- When *A* = 0 and *B* = 1, the upper pMOS is ON and lower pMOS is OFF. So, output cannot be connected to *V*DD. Under this condition, left nMOS is OFF and right nMOS is ON. Hence, the output is connected to the ground and we get logic low at the output.
- When *A* = 1 and *B* = 1, both nMOS transistors are ON and both pMOS transistors are OFF. Hence, the output is connected to *V*DD and we get logic low at the output. This proves the truth table of NOR gate as shown in Fig. 6.14(b).



NOR gate in CMOS

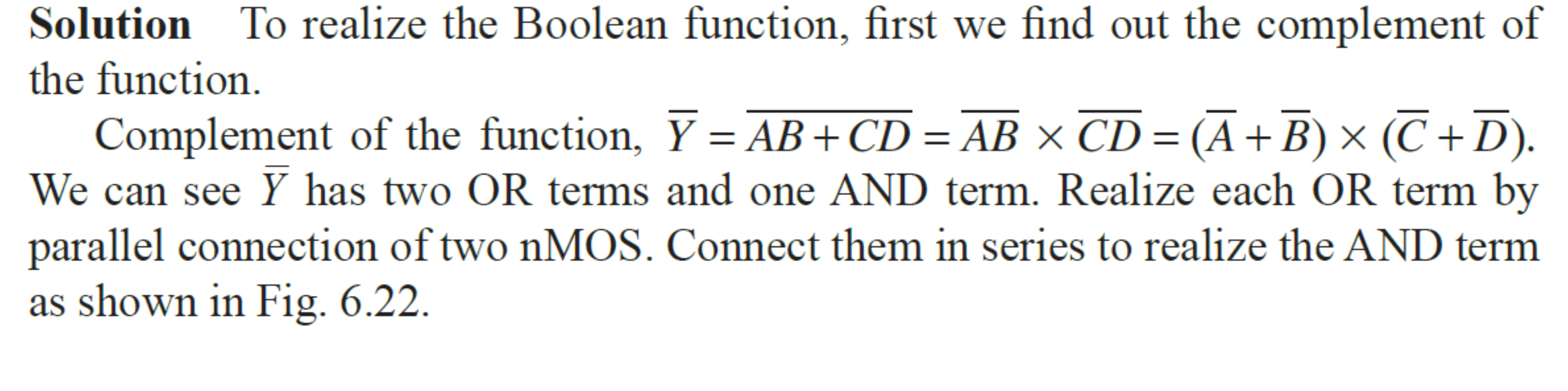


Design the circuit with a suitable stick diagram

$$Y = AB + CD$$

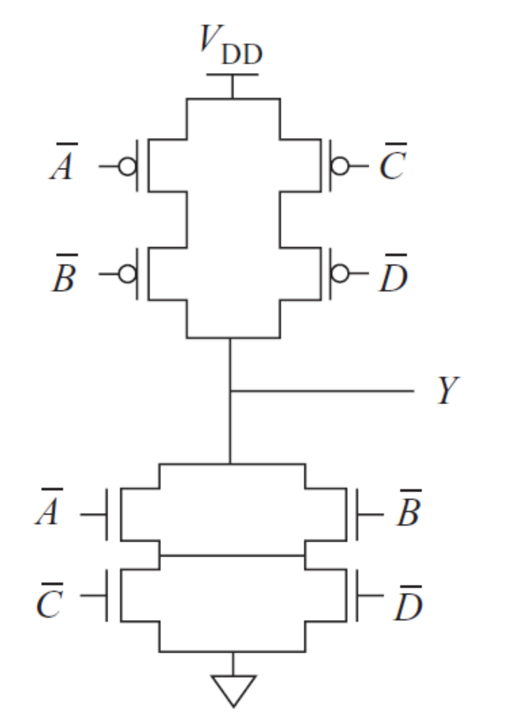
$$Y = \overline{A(B+C)+DE}$$

Solution



Solution To realize the Boolean function, first we find out the complement of the function.

Complement of the function, $\overline{Y} = \overline{AB + CD} = \overline{AB} \times \overline{CD} = (\overline{A} + \overline{B}) \times (\overline{C} + \overline{D})$. We can see $\overline{Y}$ has two OR terms and one AND term. Realize each OR term by parallel connection of two nMOS. Connect them in series to realize the AND term as shown in Fig. 6.22.



Solution

Step 1: Take complement of *Y*

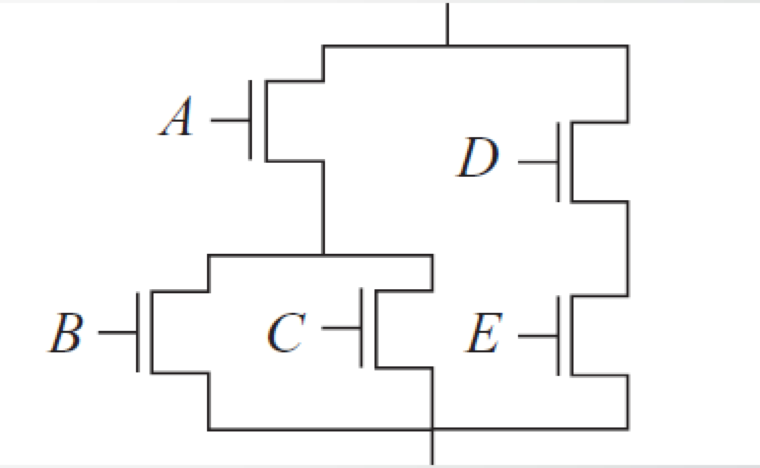
$$\overline{Y} = \overline{\overline{A(B+C)+DE}} = A(B+C)+DE$$

Step 2: Design of PDN

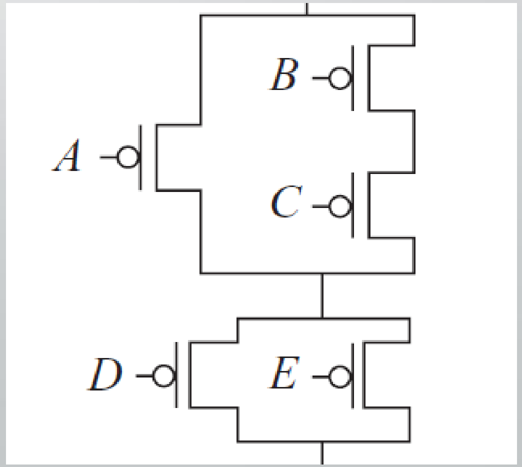
Let $Z =$ B+C , then Eqn (6.6) becomes

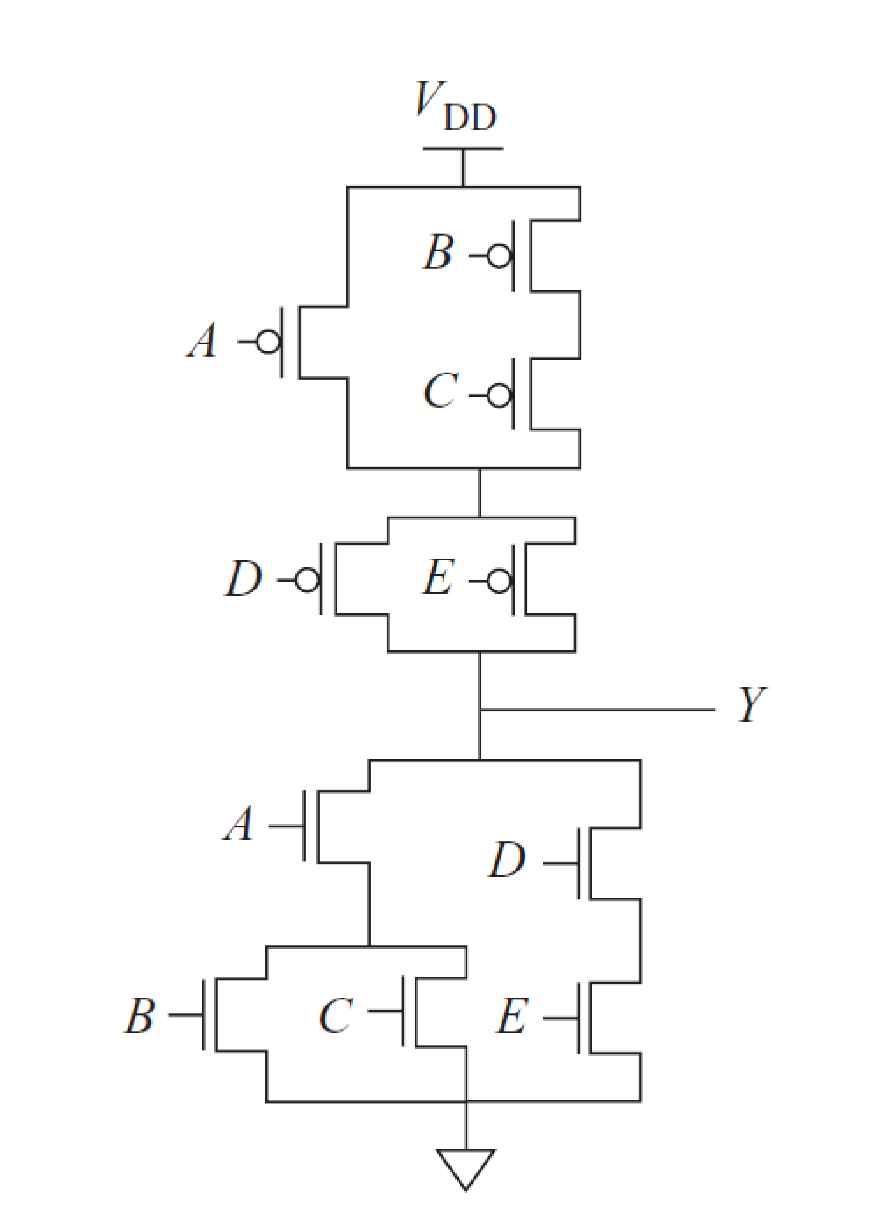
$$\overline{Y} = AZ + DE$$

Now, Z(= B+C) is realized by two nMOSFETs connected in parallel.



Step 3: Design of PUN




V_{DD}
B
A
C
D
E
Y
A
D
B
C
E

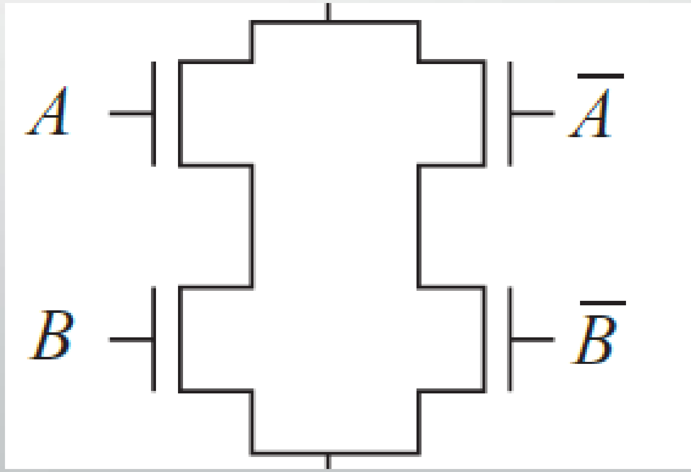
Design of XOR Gate

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

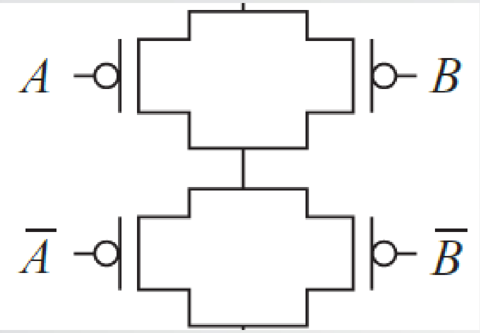
Step 1: Take complement of *Y*

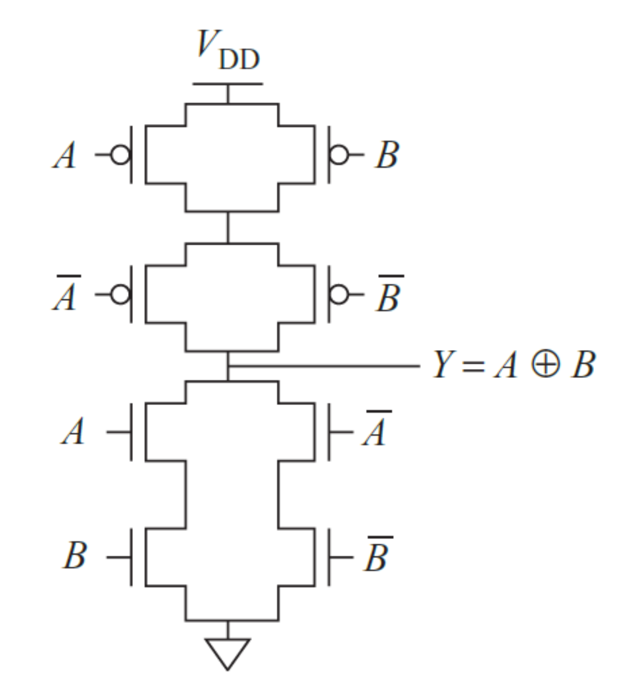
$$\overline{Y} = \overline{A\overline{B} + \overline{A}B} = \overline{A\overline{B}} \times \overline{\overline{A}B} = (\overline{A} + B) \times (A + \overline{B}) = \overline{A}\,\overline{B} + AB$$

Step 2: Design of PDN

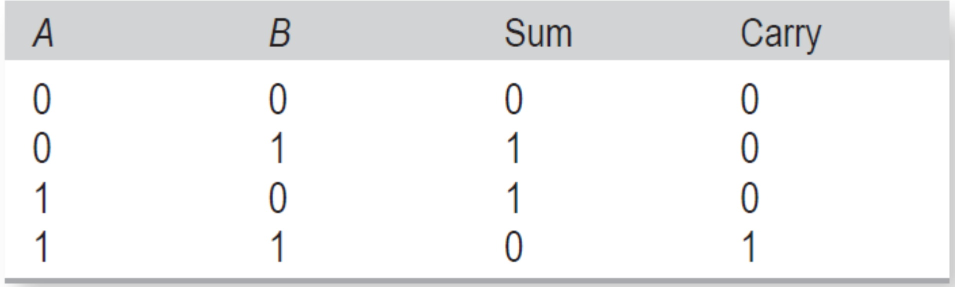


Step 3: Design of PUN





Design of Half-adder Circuit

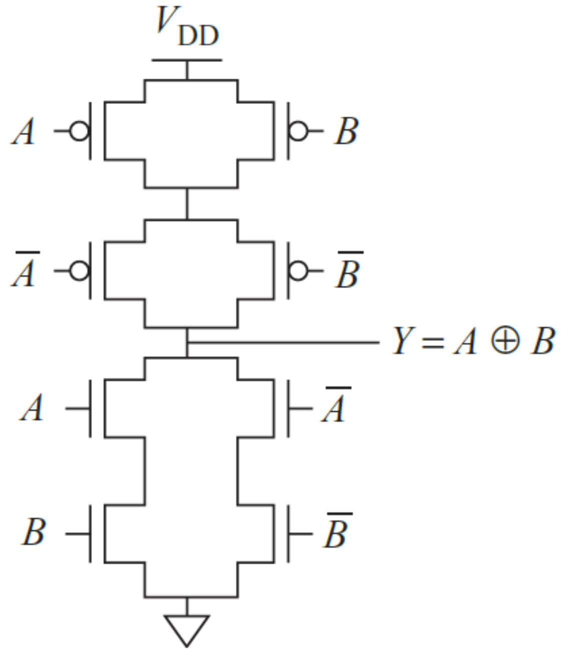


A	*B*	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

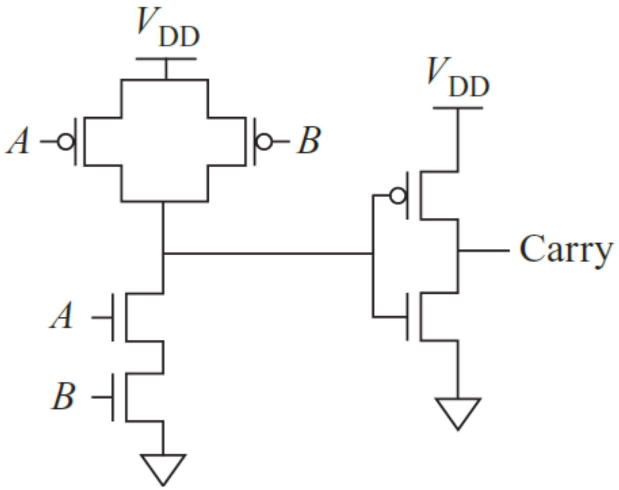
$$\text{Sum} = A\overline{B} + \overline{A}B$$

$$\text{Carry} = AB$$

sum



Carry



Design of Full-adder Circuit

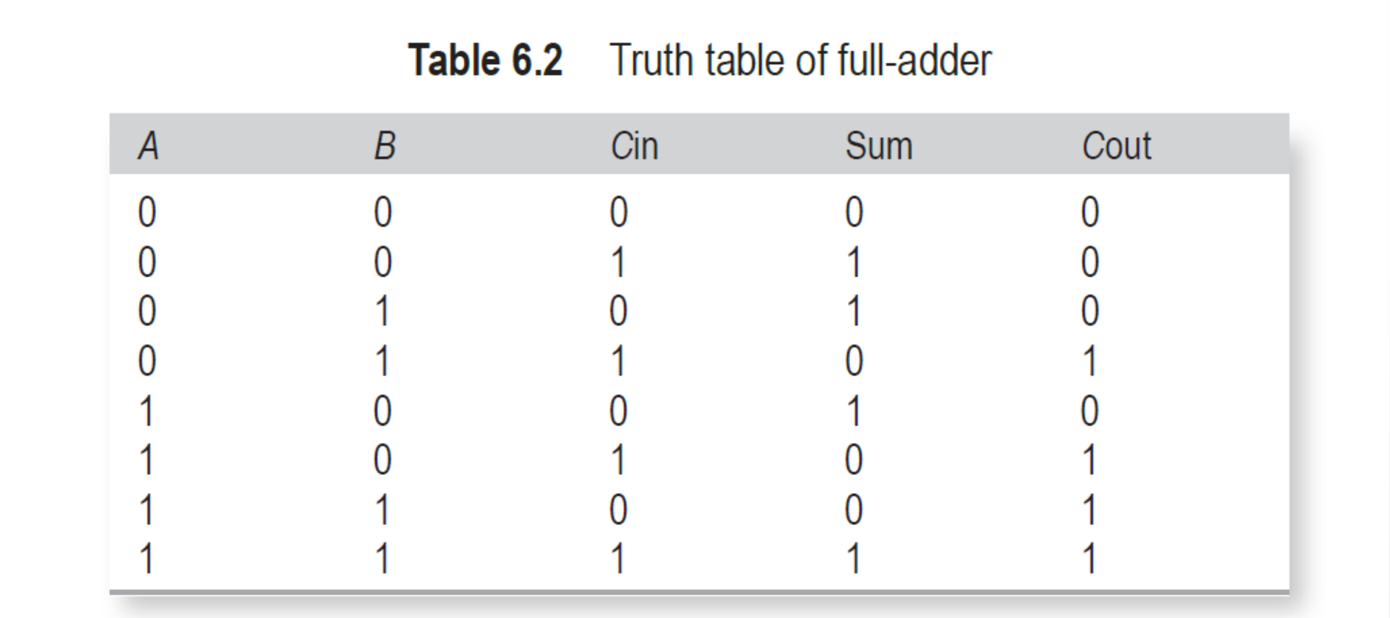
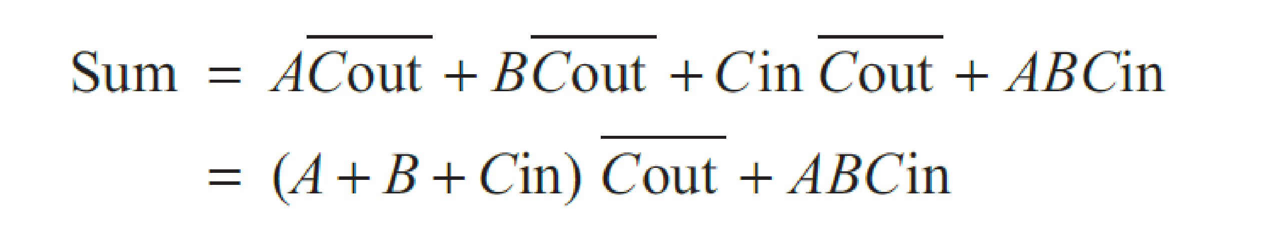


Table 6.2 Truth table of full-adder

A	*B*	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\text{Sum} = \overline{A}\,\overline{B}C\text{in} + \overline{A}B\overline{C\text{in}} + A\overline{B}\,\overline{C\text{in}} + ABC\text{in}$$

$$\text{Cout} = AB + AC\text{in} + BC\text{in}$$



$$\begin{aligned}\text{Sum} &= A\overline{\text{Cout}} + B\overline{\text{Cout}} + C\text{in}\,\overline{\text{Cout}} + ABC\text{in}\\ &= (A + B + C\text{in})\,\overline{\text{Cout}} + ABC\text{in}\end{aligned}$$

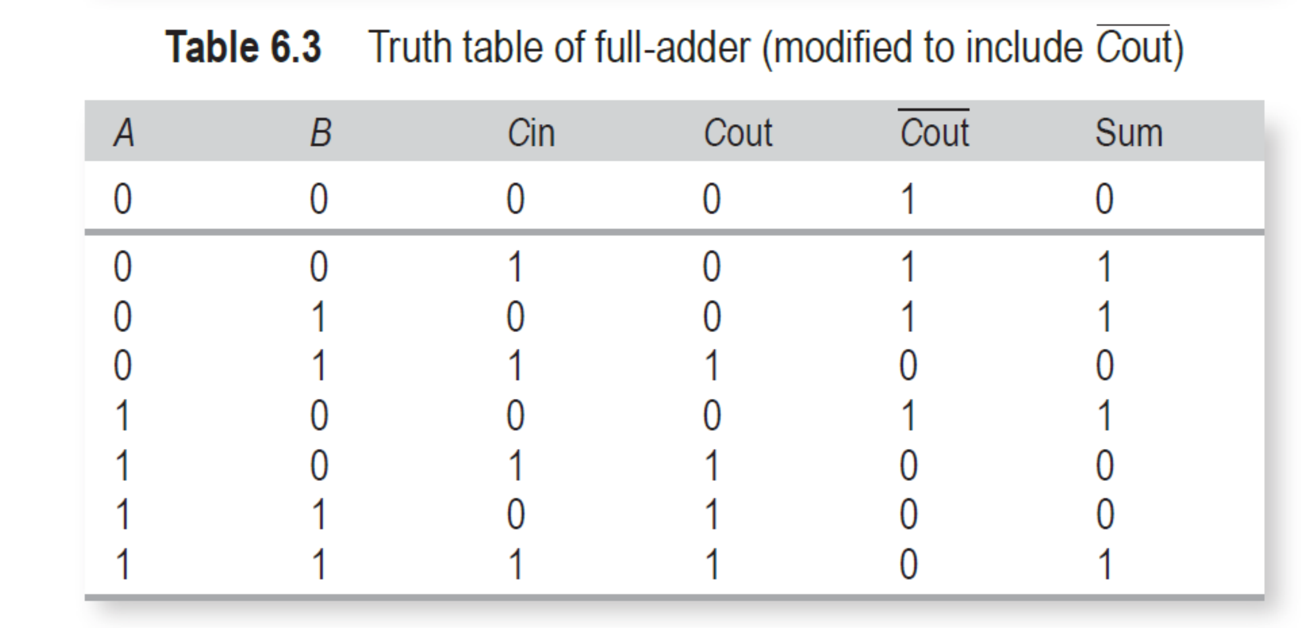
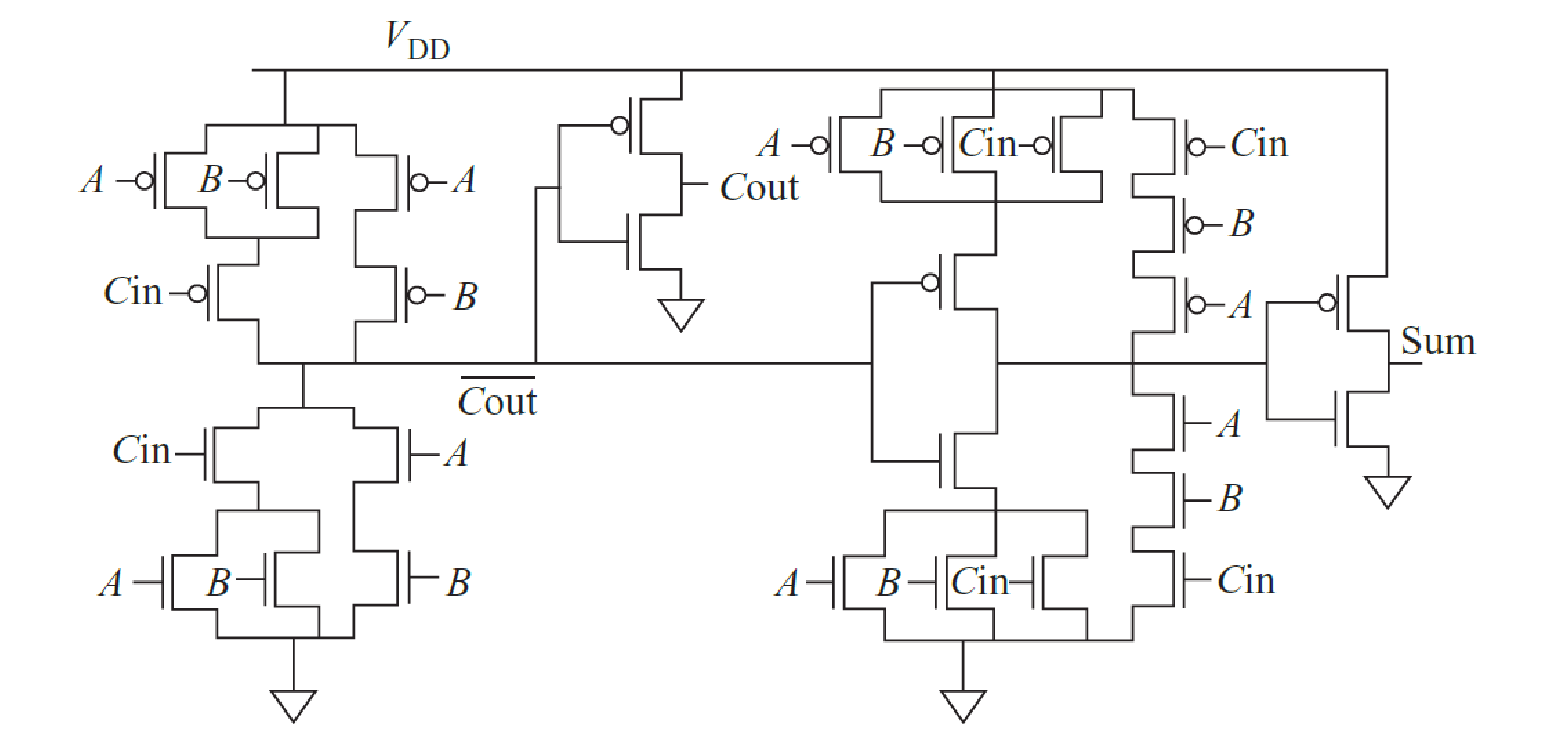


Table 6.3 Truth table of full-adder (modified to include $\overline{\text{Cout}}$)

A	*B*	Cin	Cout	$\overline{\text{Cout}}$	Sum
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	0	1


V_{DD}
A
B
A
Cin
B
Cout
$\overline{\text{Cout}}$
Cin
A
A
B
B
A
B
Cin
Cin
B
A
Sum
A
B
A
B
Cin
Cin