

---

# Logic Design with MOSFETs

Dae Hyun Kim

EECS

Washington State University

---

# References

---

- John P. Uyemura, “Introduction to VLSI Circuits and Systems,” 2002.
  - Chapter 2
- Neil H. Weste and David M. Harris, “CMOS VLSI Design: A Circuits and Systems Perspective,” 2011.
  - Chapter 1

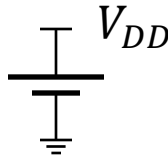
# Goal

---

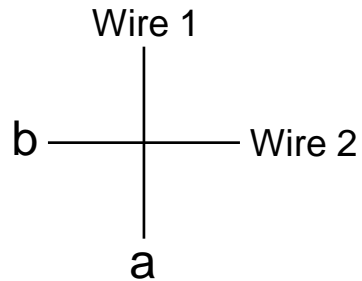
- Design logic gates using MOSFETs (NMOS and PMOS)

# Signals and Wires

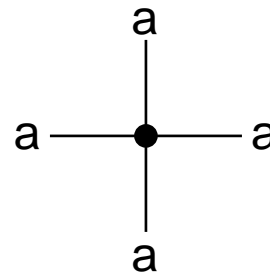
- Signals
  - $0 = V_{SS} = \text{Ground} = \text{GND} = \text{Low} = 0\text{V}$
  - $1 = V_{DD} = \text{Power} = \text{PWR} = \text{High} = 5\text{V}, 3.3\text{V}, 1.5\text{V}, 1.2\text{V}, 1.0\text{V}, \text{etc.}$



- Wires



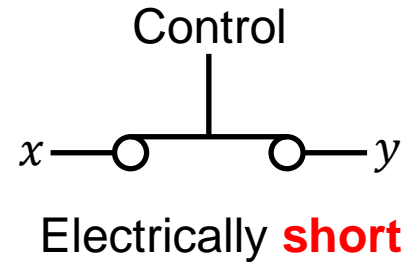
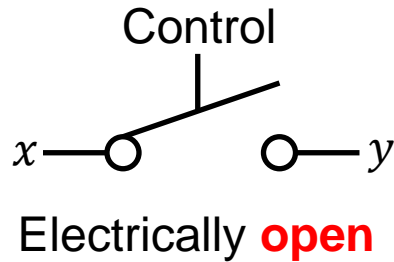
No connection



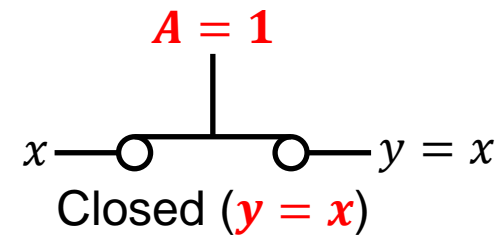
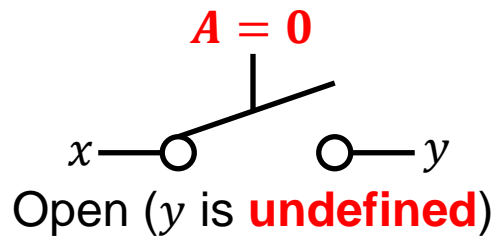
Connection

# Ideal Switches

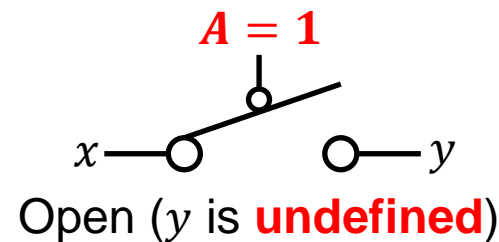
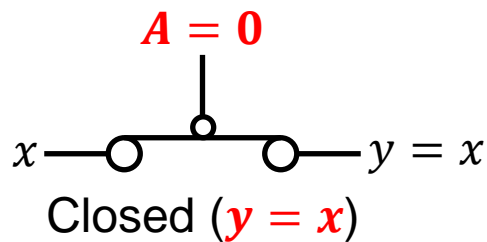
- Switch



- Assert-high switch

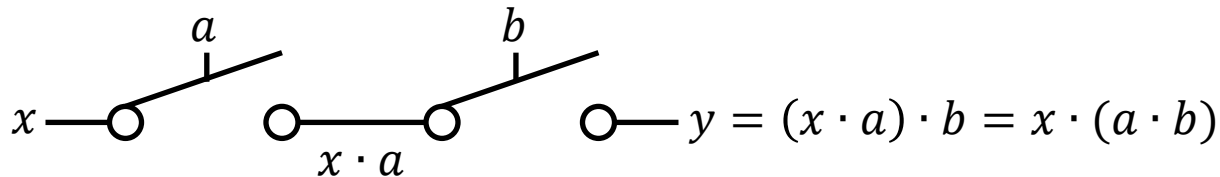


- Assert-low switch



# Series/Parallel Connections of Switches

- Series



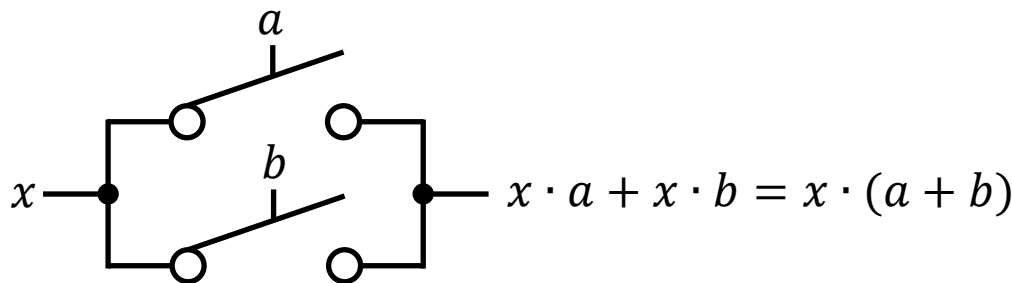
AND operation

( $y$  is defined only when  $a = 1$  and  $b = 1$ )

( $y$  is undefined if  $a = 0$  or  $b = 0$ )

a	b	y
0	0	undefined
0	1	
1	0	
1	1	x

- Parallel



OR operation

( $y$  is defined only when  $a = 1$  or  $b = 1$ )

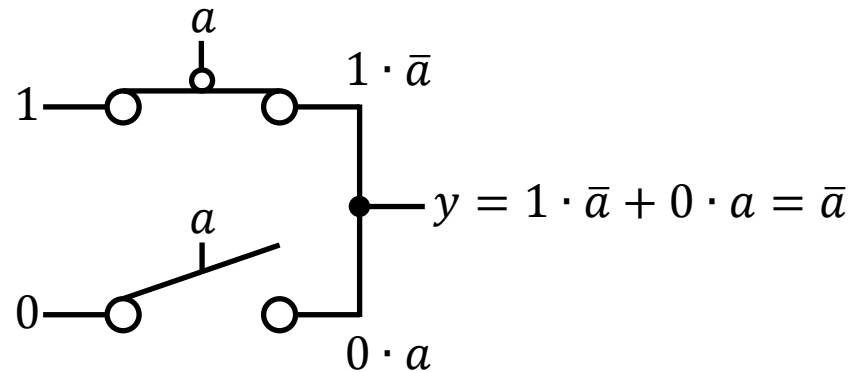
( $y$  is undefined if  $a = 0$  and  $b = 0$ )

a	b	y
0	0	undefined
0	1	x
1	0	
1	1	

# Inverter Design with Switches

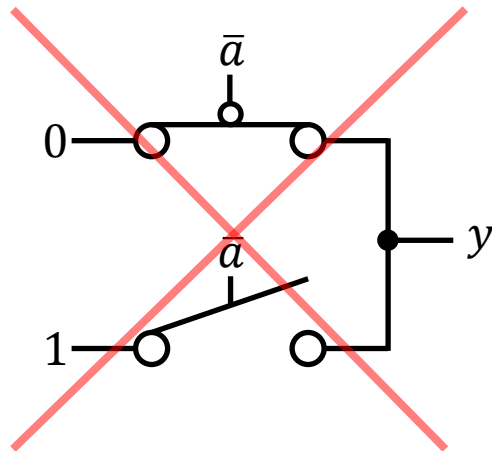
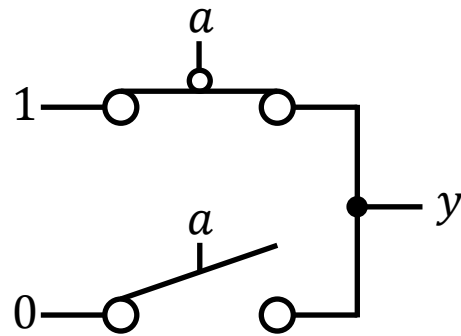
- Inverter
  - The output is defined both when  $a = 0$  and when  $a = 1$ .

$a$	$y$
0	1
1	0



# Inverter Design with Switches

- Two inverter designs



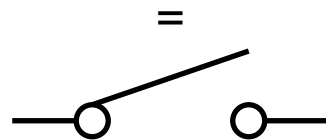
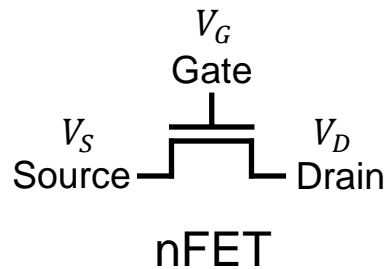
Why?



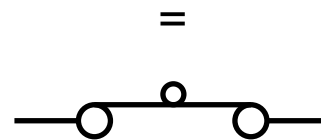
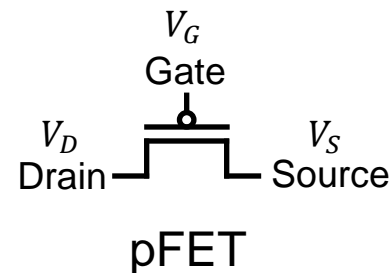
# MOSFETs as Switches

- MOSFET: Metal-Oxide-Semiconductor Field-Effect Transistor
  - n-channel MOSFET = nFET = NMOS
  - p-channel MOSFET = pFET = PMOS
  - Complementary MOS: CMOS

- Symbols



$$(V_D \geq V_S)$$



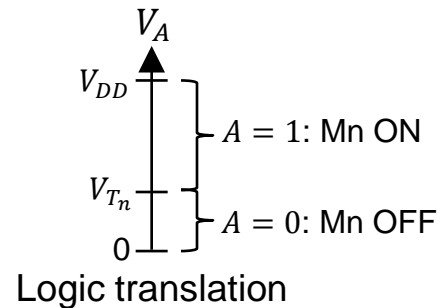
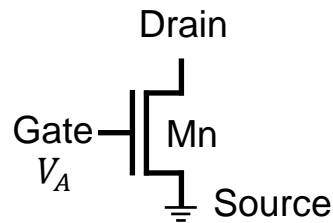
$$(V_S \geq V_D)$$

# MOSFETs as Switches

- Threshold voltage
  - nFET:  $V_{Tn} > 0$
  - pFET:  $V_{Tp} < 0$

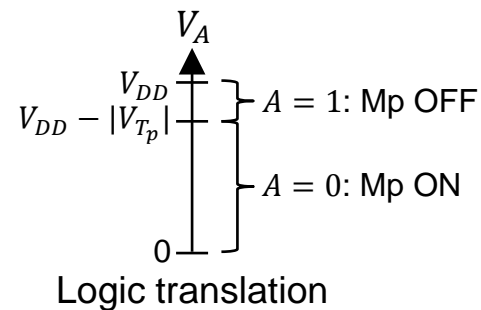
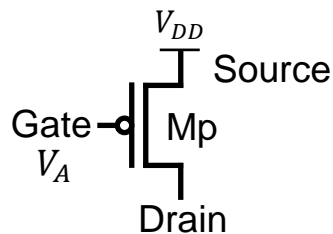
- nFET

- OFF:  $V_{GSn} \leq V_{Tn}$
- ON:  $V_{GSn} > V_{Tn}$



- pFET

- OFF:  $V_{SGp} \leq |V_{Tp}|$
- ON:  $V_{SGp} > |V_{Tp}|$



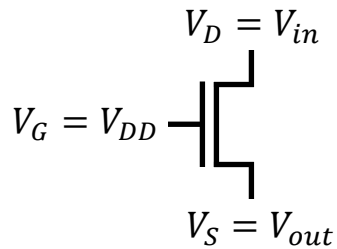
# MOSFETs as Switches

---

- Example (PTM High-Performance 45nm High-K Metal Gate)
  - $V_{DD}$ : 1.0V
  - $V_{T_n}$ : 0.46893V
  - $V_{T_p}$ : -0.49158V
- Example (PTM High-Performance 32nm High-K Metal Gate)
  - $V_{DD}$ : 0.9V
  - $V_{T_n}$ : 0.49396V
  - $V_{T_p}$ : -0.49155V
- Example (PTM High-Performance 22nm High-K Metal Gate)
  - $V_{DD}$ : 0.8V
  - $V_{T_n}$ : 0.50308V
  - $V_{T_p}$ : -0.4606V

# Pass Characteristics

- nFET

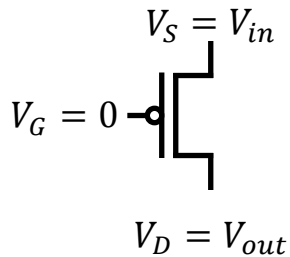


$V_{in} \uparrow$	$V_{GS} \downarrow$	$V_{out} \uparrow$
0	$V_{DD}$	0
0.1	$V_{DD} - 0.1$	0.1
...	...	...
$V_{DD} - V_{Tn}$	$V_{Tn}$	$V_{DD} - V_{Tn}$
$V_{DD}$	$V_{Tn}$	$V_{DD} - V_{Tn}$

Logic 0 transfer: **strong logic 0**

Logic 1 transfer: **weak logic 1**

- pFET



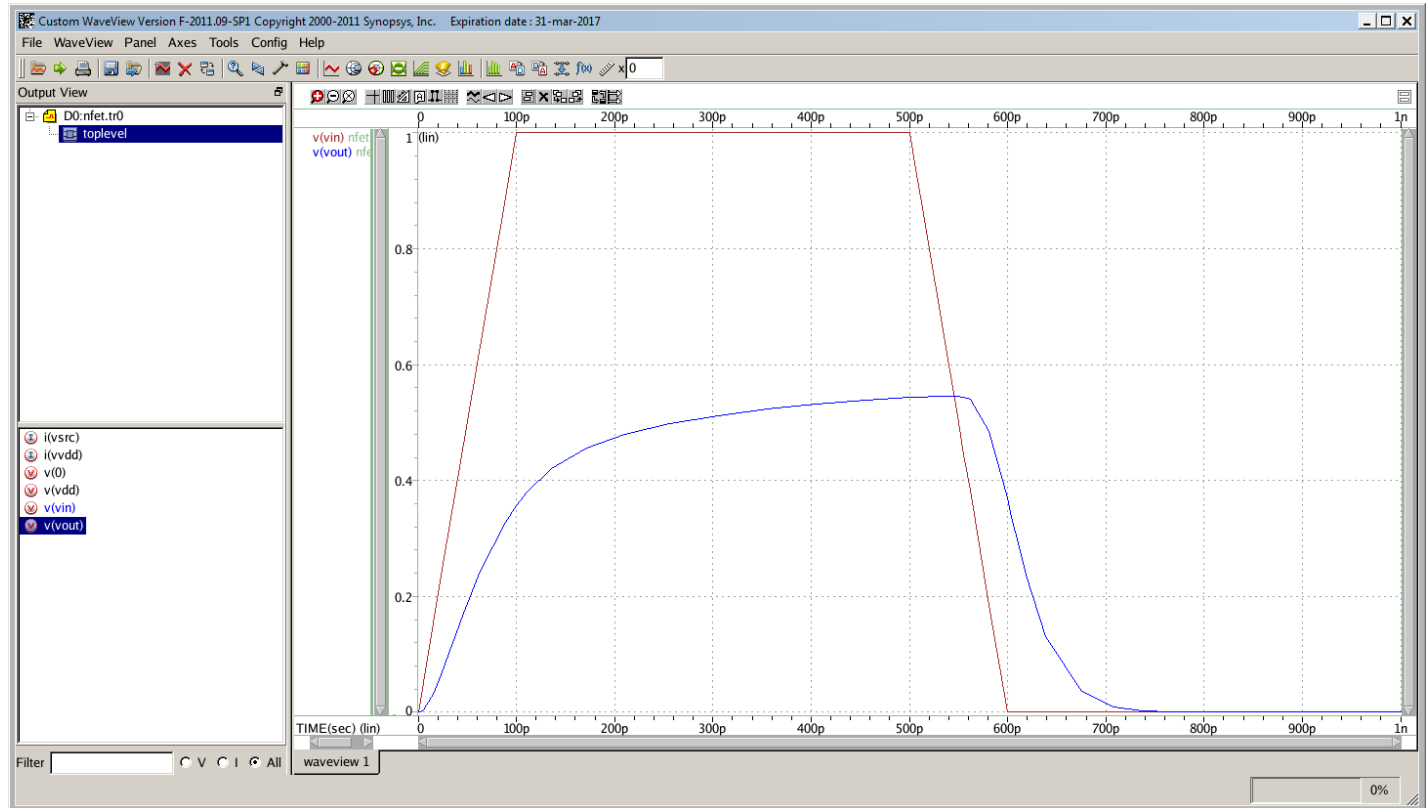
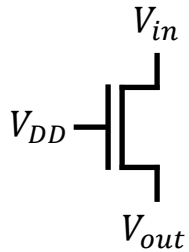
$V_{in} \downarrow$	$V_{SG} \downarrow$	$V_{out} \downarrow$
$V_{DD}$	$V_{DD}$	$V_{DD}$
$V_{DD} - \epsilon$	$V_{DD} - \epsilon$	$V_{DD} - \epsilon$
...	...	...
$ V_{Tp} $	$ V_{Tp} $	$ V_{Tp} $
0	0	$ V_{Tp} $

Logic 1 transfer: **strong logic 1**

Logic 0 transfer: **weak logic 0**

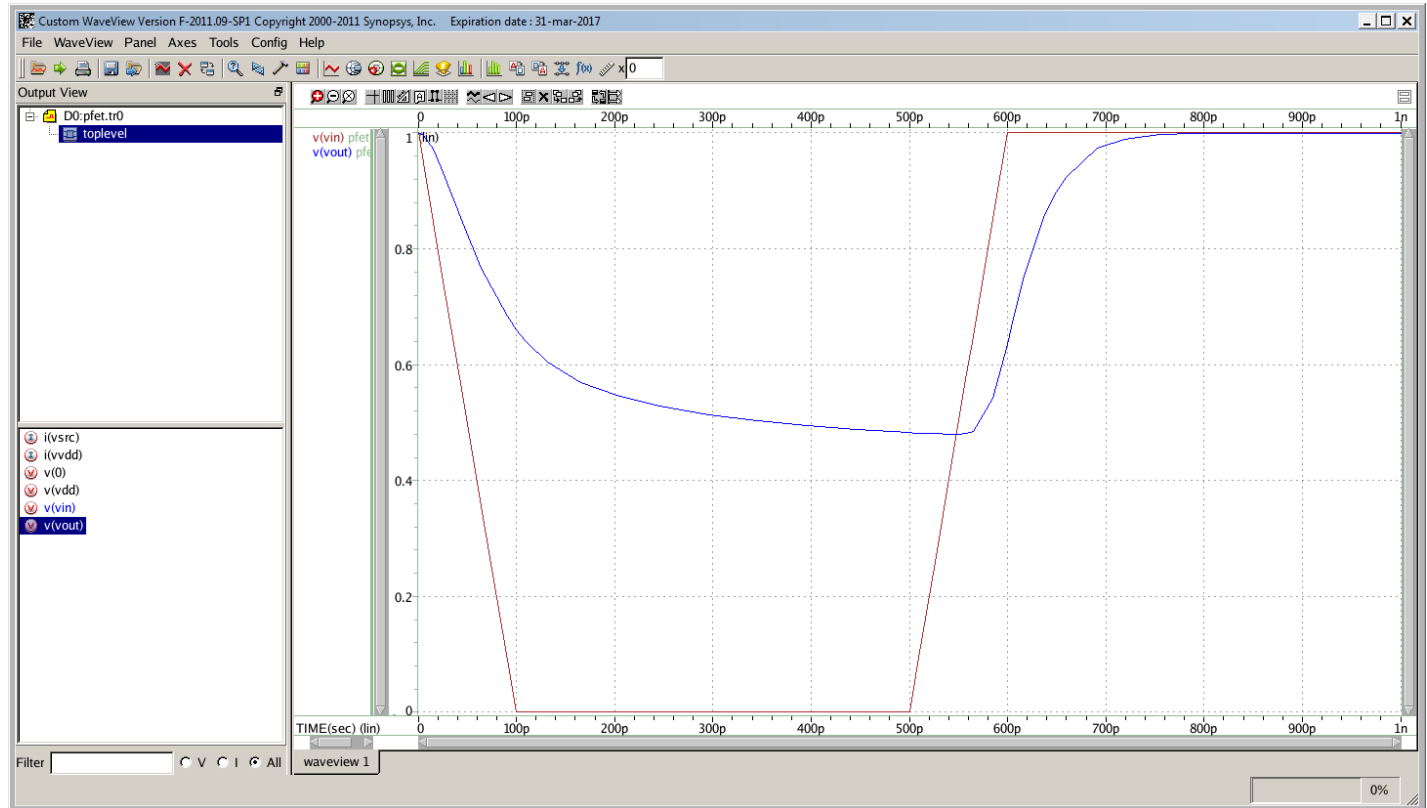
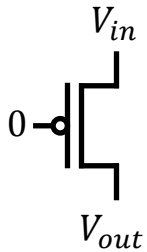
# Pass Characteristics

- SPICE simulation (45nm technology)
  - nFET



# Pass Characteristics

- SPICE simulation (45nm technology)
  - pFET



# Pass Characteristics

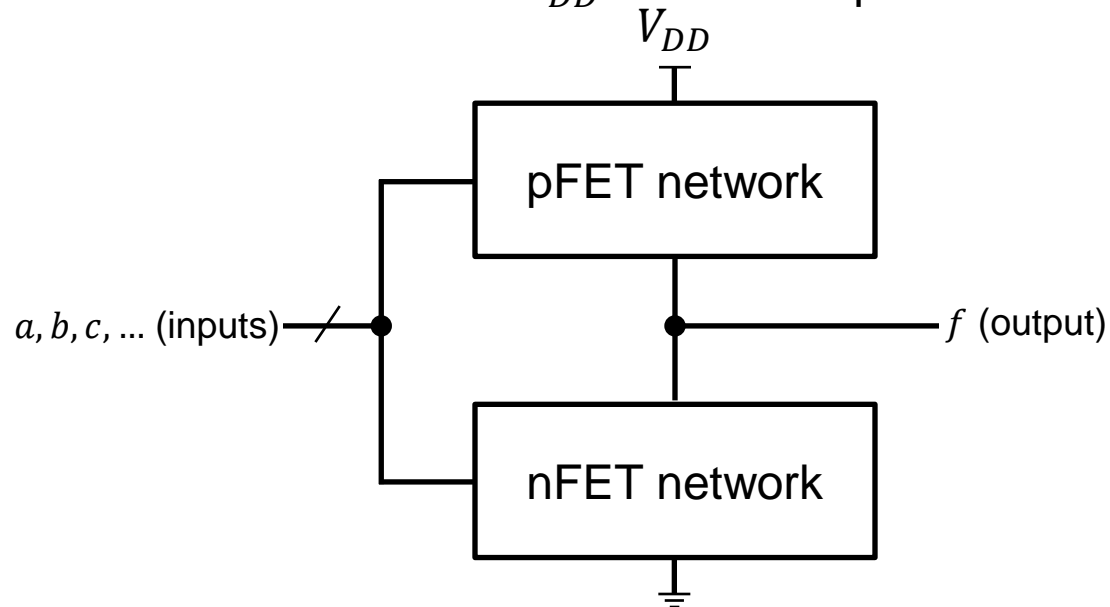
---

- nFET
  - **Strong logic 0** transfer
  - Weak logic 1 transfer
- pFET
  - **Strong logic 1** transfer
  - Weak logic 0 transfer
- CMOS
  - Use pFETs to pass logic 1.
  - Use nFETs to pass logic 0.

# Basic Logic Gates in CMOS

- Principles

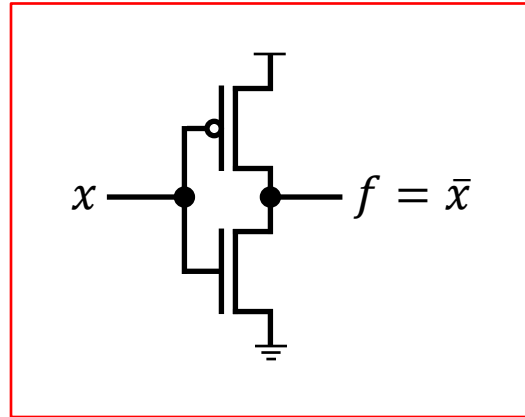
- Construct the nFET network using only nFETs and the pFET network using only pFETs.
- If the output is 1, the pFET network connects  $V_{DD}$  to the output and the nFET network disconnects  $V_{SS}$  and the output.
- If the output is 0, the nFET network connects  $V_{SS}$  to the output and the pFET network disconnects  $V_{DD}$  and the output.



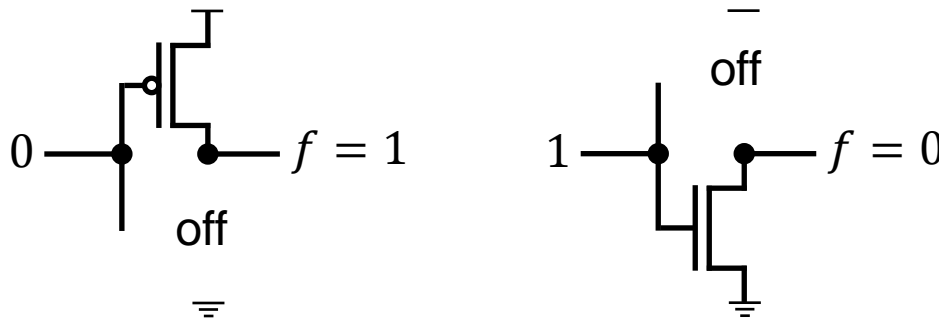


# Basic Logic Gates in CMOS

- Inverter



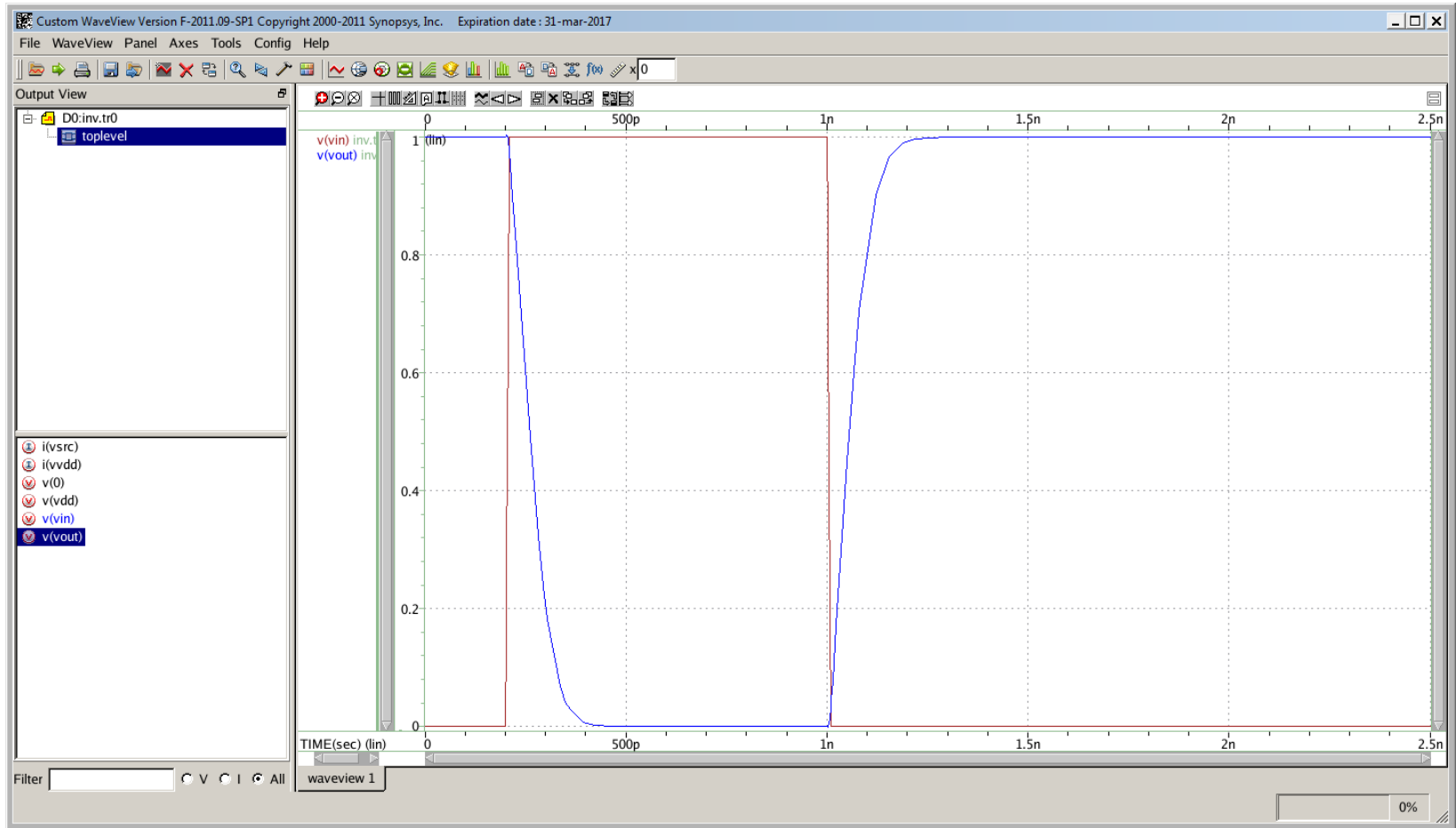
# TRs: 2  
nFET: 1  
pFET: 1



$$f = \bar{x} \cdot 1 + x \cdot 0 = \bar{x}$$

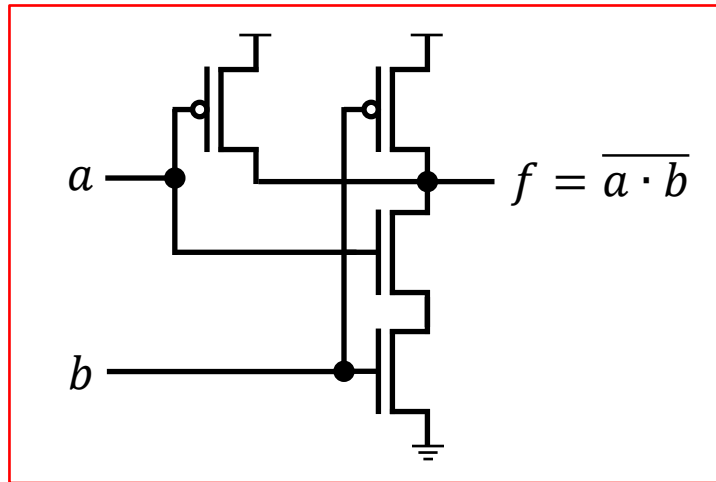
# Basic Logic Gates in CMOS

- SPICE simulation

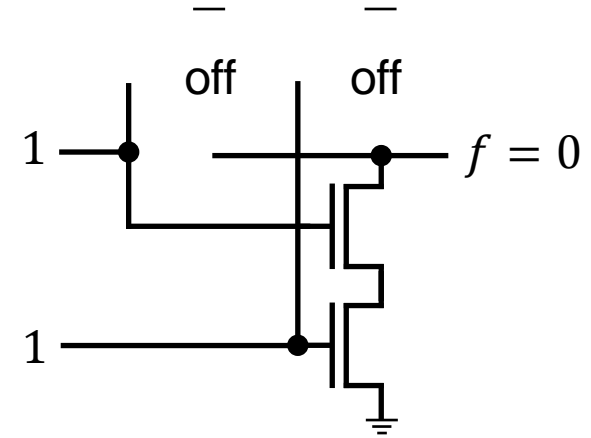
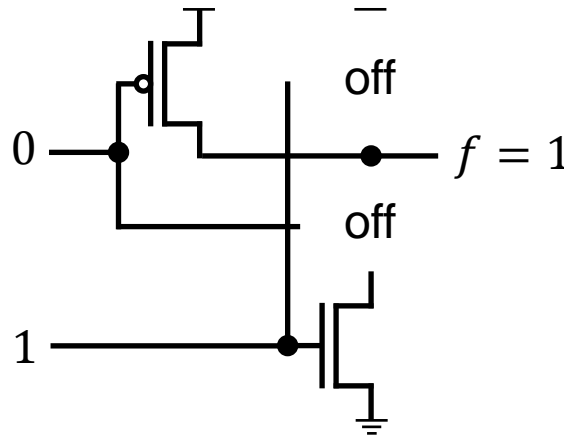
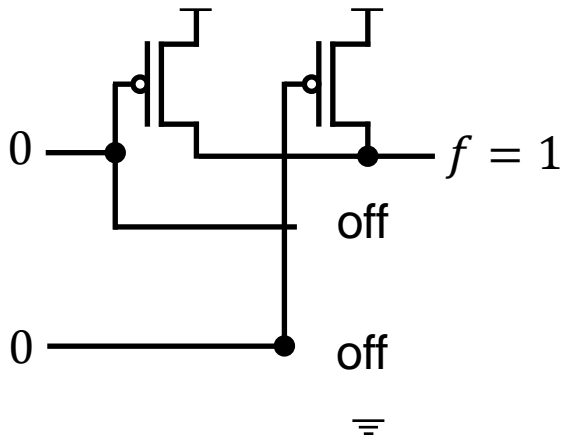


# Basic Logic Gates in CMOS

- Two-input NAND (NAND2)



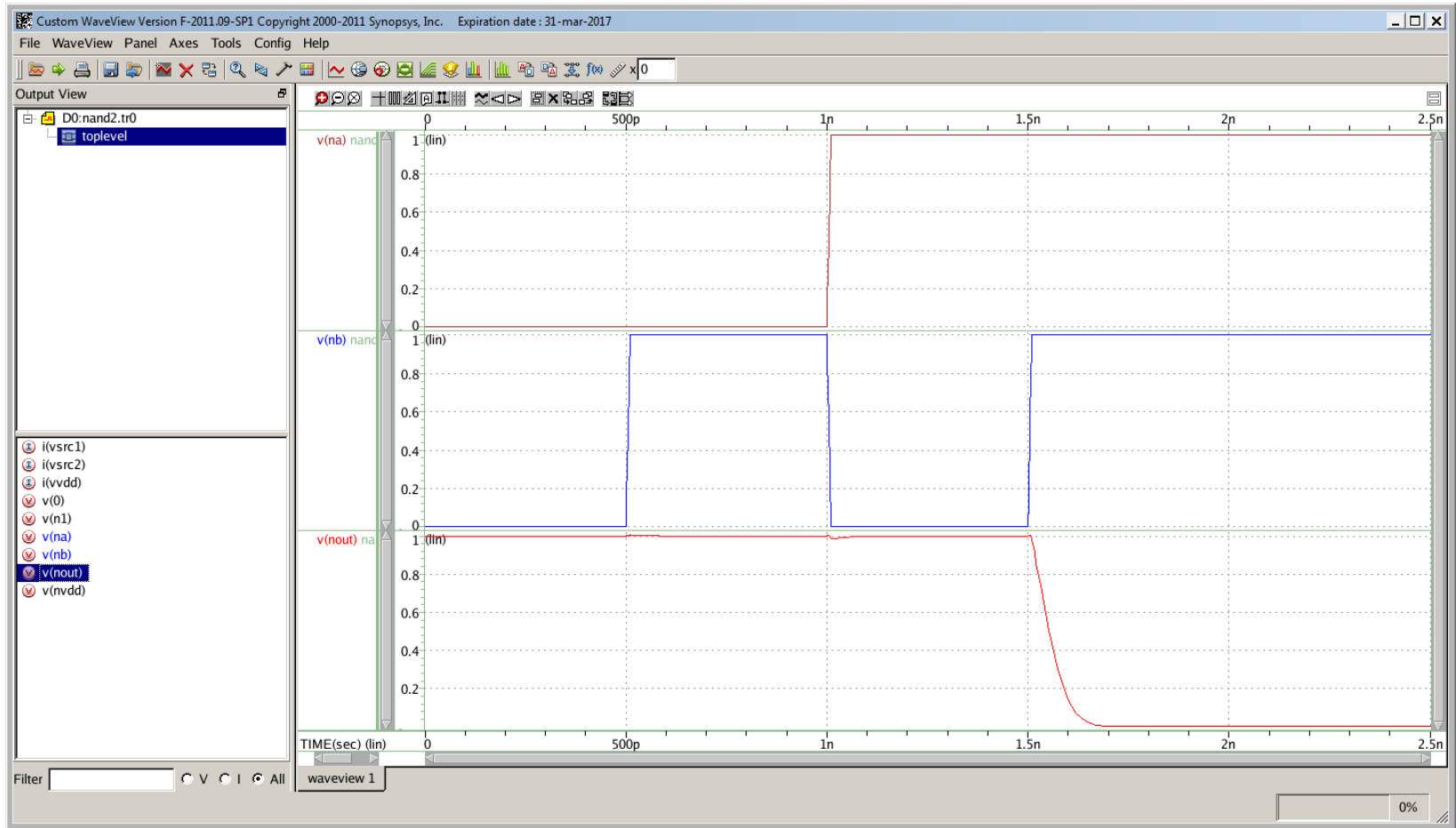
# TRs: 4  
nFETs: 2  
pFETs: 2



$$f = \bar{a} \cdot \bar{b} \cdot 1 + \bar{a} \cdot b \cdot 1 + a \cdot \bar{b} \cdot 1 + a \cdot b \cdot 0 = \bar{a} + \bar{b} = \overline{a \cdot b}$$

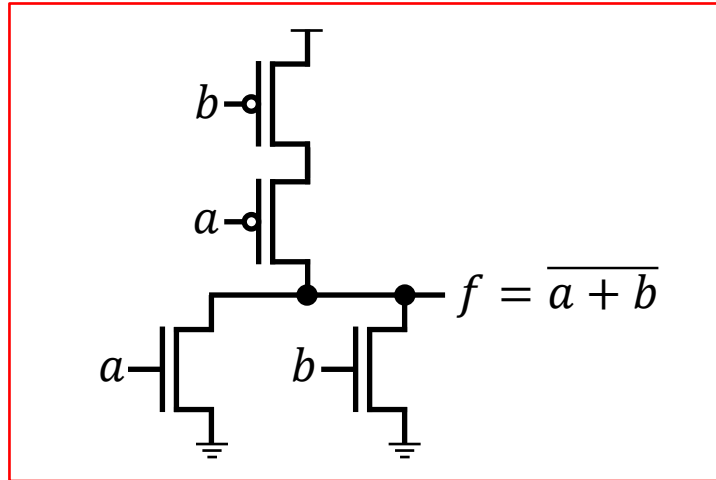
# Basic Logic Gates in CMOS

- SPICE simulation

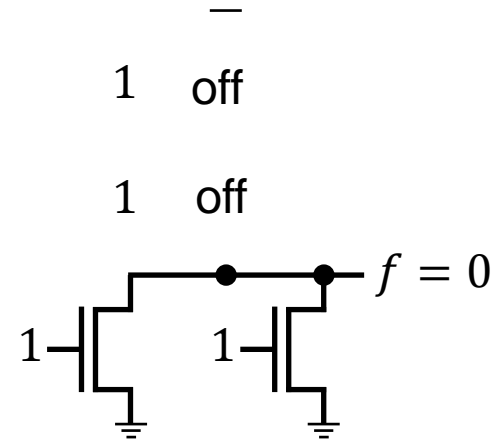
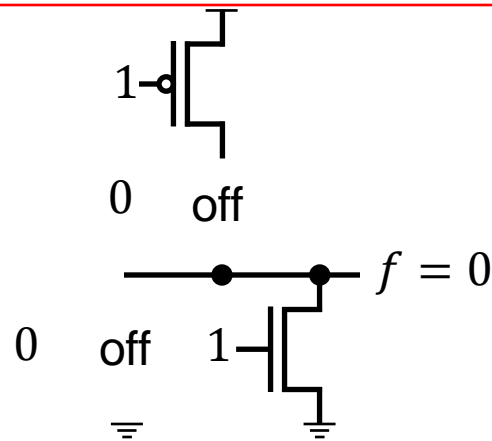
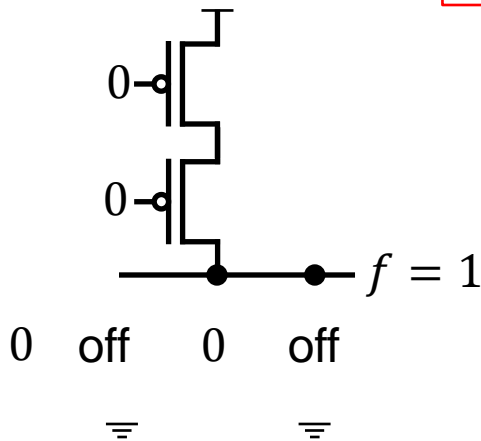


# Basic Logic Gates in CMOS

- Two-input NOR (NOR2)



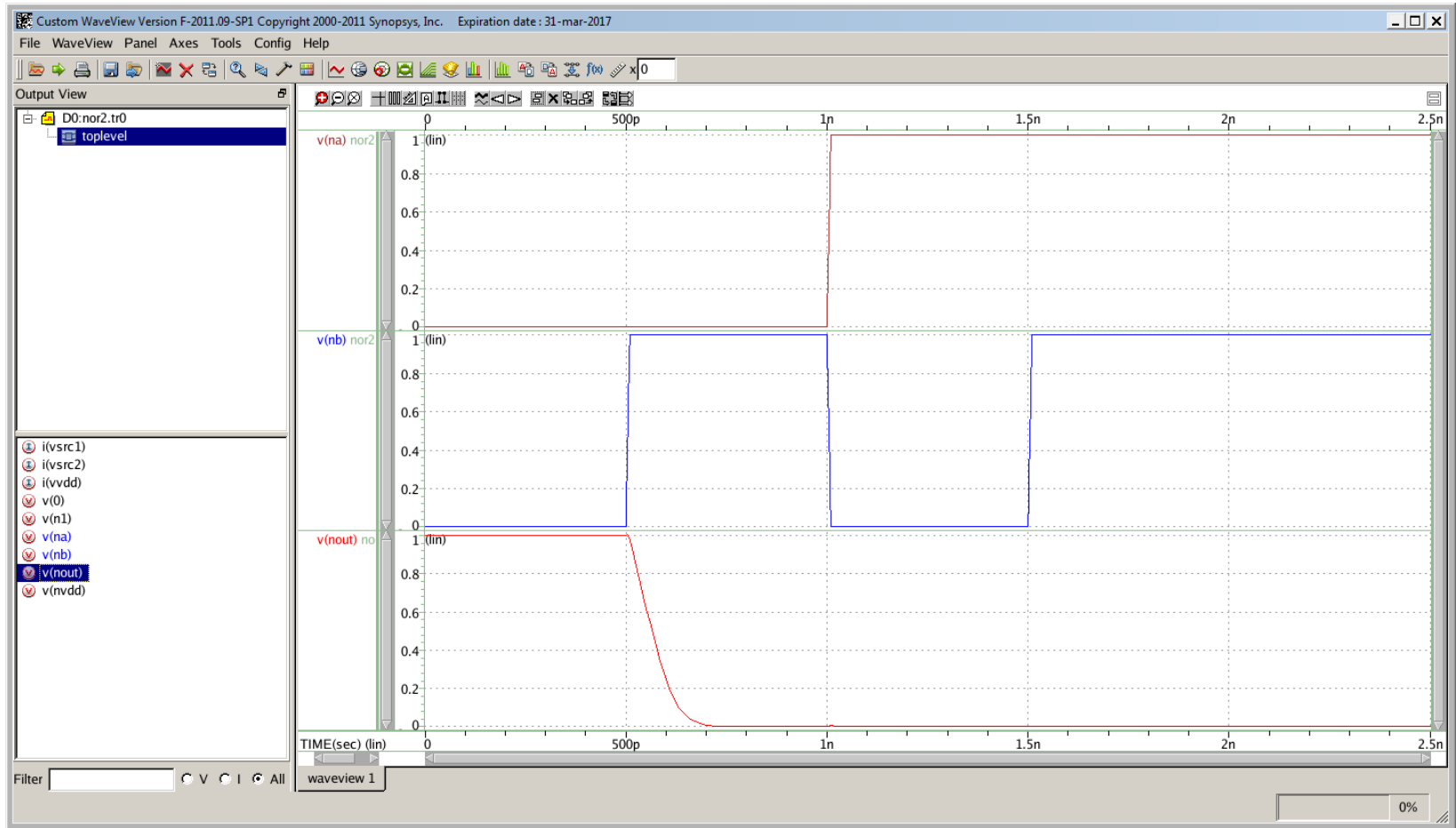
# TRs: 4  
nFETs: 2  
pFETs: 2



$$f = \bar{a} \cdot \bar{b} \cdot 1 + \bar{a} \cdot b \cdot 0 + a \cdot \bar{b} \cdot 0 + a \cdot b \cdot 0 = \bar{a} \cdot \bar{b} = \overline{a + b}$$

# Basic Logic Gates in CMOS

- SPICE simulation

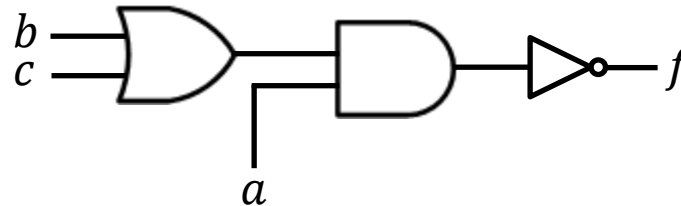


# Complex Logic Gates in CMOS

- Example

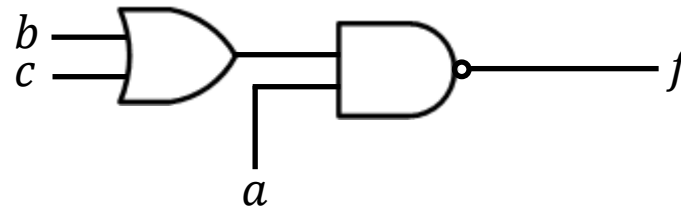
$$f = \overline{a \cdot (b + c)}$$

- Using logic gates



# TRs: 14  
nFETs: 7  
pFETs: 7

- Using logic gates



# TRs: 10  
nFETs: 5  
pFETs: 5

- Using TRs

# TRs: 6  
nFETs: 3  
pFETs: 3

# Complex Logic Gates in CMOS

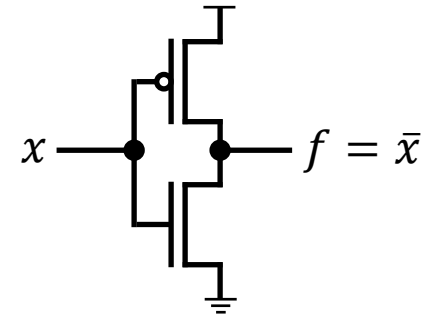
- How to design

- Inverter

$$f = \bar{x} = \boxed{\bar{x}} \cdot 1 + \boxed{x} \cdot 0$$

pFET network  
(connects 1 and the output)

nFET network  
(connects 0 and the output)

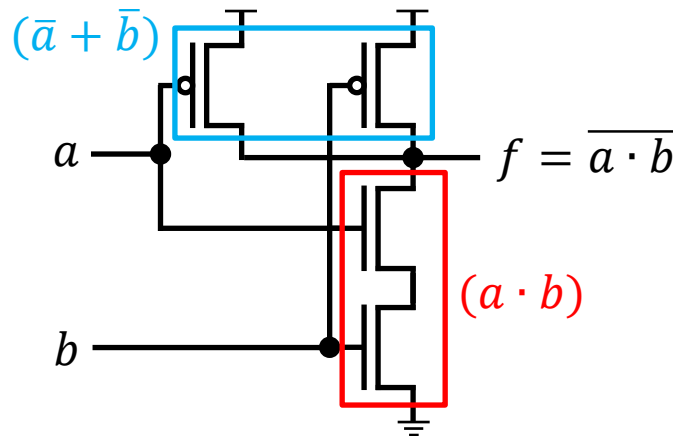


- NAND2

$$f = \overline{a \cdot b} = (\overline{a \cdot b}) \cdot 1 + (a \cdot b) \cdot 0 = \boxed{(\bar{a} + \bar{b})} \cdot 1 + \boxed{(a \cdot b)} \cdot 0$$

pFET network  
(expressed by  $\bar{a} + \bar{b}$ )

nFET network  
(expressed by  $a \cdot b$ )





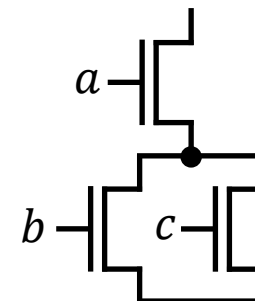
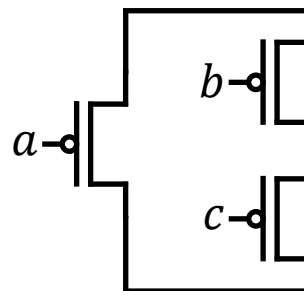
# Complex Logic Gates in CMOS

- How to design  $f$ 
  - Express  $f = A \cdot 1 + B \cdot 0 = F(\overline{x_1}, \dots, \overline{x_n}) \cdot 1 + \overline{F(x_1, \dots, x_n)} \cdot 0$
  - Design a pFET network using  $A = F(\overline{x_1}, \dots, \overline{x_n})$ .
    - pFETs are ON when the inputs are 0.
  - Design an nFET network using  $B = \overline{F(x_1, \dots, x_n)}$ .
    - nFETs are ON when the inputs are 1.

- Example

$$f = \overline{a \cdot (b + c)}$$

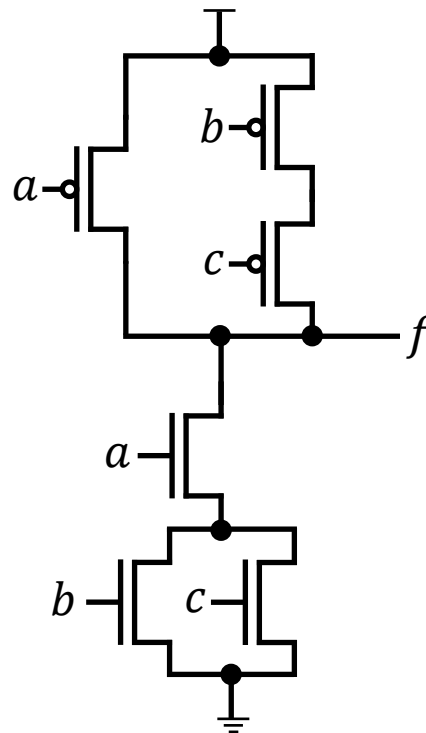
$$- f = \overline{a \cdot (b + c)} \cdot 1 + a \cdot (b + c) \cdot 0 = \underbrace{(\overline{a} + \overline{b} \cdot \overline{c})}_{\text{pFET network}} \cdot 1 + \underbrace{a \cdot (b + c)}_{\text{nFET network}} \cdot 0$$



# Complex Logic Gates in CMOS

- Example

$$f = \overline{a \cdot (b + c)}$$



# TRs: 6  
nFETs: 3  
pFETs: 3

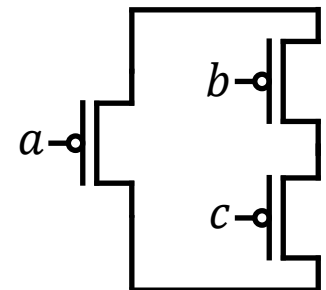
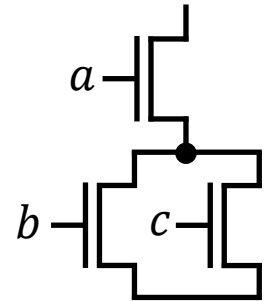
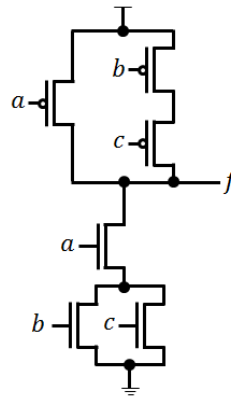
# Complex Logic Gates in CMOS

---

- Structured logic design
  - Design a given Boolean equation using nFETs and pFETs.
- Assume that only non-inverted input signals are given.
  - $a, b, c, \dots$  are given.
  - $\bar{a}, \bar{b}, \bar{c}, \dots$  are not given. If you need them, you should generate them.

# Complex Logic Gates in CMOS

- Design methodology 1
  - When  $f = \overline{S(x_1, \dots, x_n)}$  ( $S$  is a function of non-inverted variables)
    - $f = \bar{S} = \bar{S} \cdot 1 + S \cdot 0$
    - Design an nFET network for  $S$  using  $x_1, \dots, x_n$ .
    - Design a pFET network for  $\bar{S}$  using  $\bar{x}_1, \dots, \bar{x}_n$ .
    - Connect them to  $V_{DD}, V_{SS}, f$ .
  - Example:  $f = \overline{a \cdot (b + c)}$ 
    - $f = \overline{a \cdot (b + c)} = \bar{a} + \bar{b} \cdot \bar{c}$
    - Design an nFET network for  $a \cdot (b + c)$ .
    - Design a pFET network for  $\bar{a} + \bar{b} \cdot \bar{c}$ .
    - Connect them.



# Complex Logic Gates in CMOS

- Design methodology 2

- When  $f = \overline{S(x_1, \dots, x_n)}$

- $f = \bar{S} = \bar{S} \cdot 1 + S \cdot 0$

- Design an nFET network for  $S$ .

- Design a pFET network with a dual logic of the nFET network.

- Dual of  $f(x_1, \dots, x_n, 0, 1, AND, OR) = f(x_1, \dots, x_n, 1, 0, OR, AND)$

- Connect them.

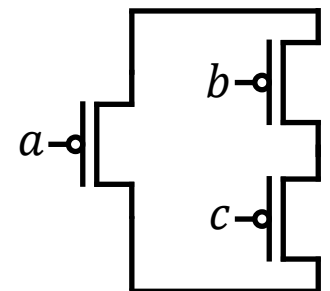
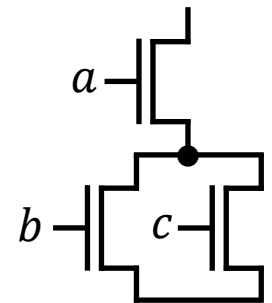
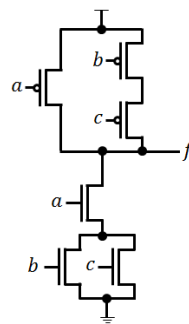
- Example:  $f = \overline{a \cdot (b + c)}$

- $f = \overline{a \cdot (b + c)} \cdot 1 + a \cdot (b + c) \cdot 0$

- Design an nFET network for  $a \cdot (b + c)$ .

- Dual of  $a \cdot (b + c) = a + (b \cdot c) = a + b \cdot c$ .

- Connect them.



# Complex Logic Gates in CMOS

- Dual logic
  - $f(x_1, \dots, x_n, 0, 1, AND, OR)^D = f(x_1, \dots, x_n, 1, 0, OR, AND)$
  - Example
    - $(A \cdot B)^D = A + B$
    - $(A + B)^D = A \cdot B$
    - $(1 \cdot A)^D = 0 + A = A$
    - $(1 + A)^D = 0 \cdot A = 0$
    - $(0 \cdot A)^D = 1 + A = 1$
    - $(0 + A)^D = 1 \cdot A = A$
- Principles of the dual logic
  - The nFET and the pFET networks work complementarily.
  - If the nFET network is ON (i.e., connects  $V_{SS}$  to the output), the pFET network is OFF (i.e., disconnect the output from  $V_{DD}$ ) and vice versa.
  - If two networks are dual, they work complementarily.
    - Prove!

# Complex Logic Gates in CMOS

- Principles of the dual logic

- $f = \overline{S(x_1, \dots, x_n)} = f = \overline{S(x_1, \dots, x_n)} \cdot 1 + S(x_1, \dots, x_n) \cdot 0$

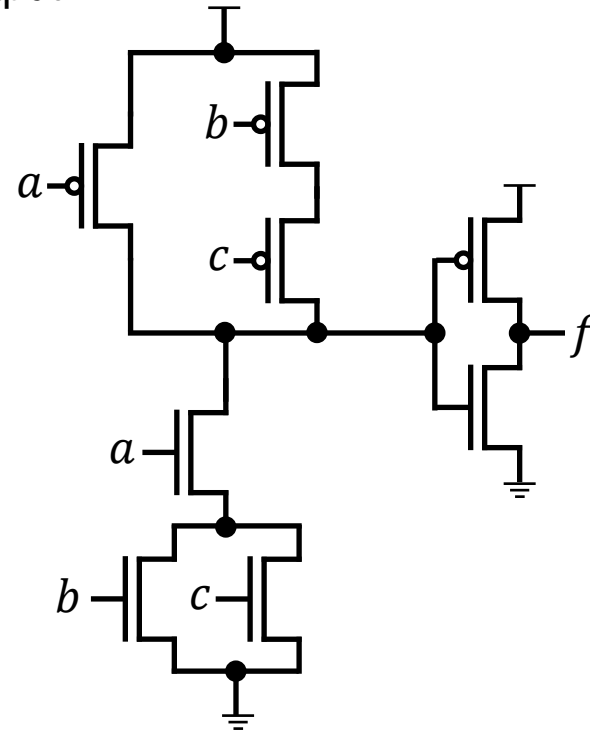
- $\overline{S(x_1, \dots, x_n)} = \overline{S(x_1, \dots, x_n, 0, 1, AND, OR)} = S(\overline{x_1}, \dots, \overline{x_n}, 1, 0, OR, AND) = S(\overline{x_1}, \dots, \overline{x_n})^D$  (De Morgan's law)

- A pFET is ON when its control variable ( $x_i$ ) is 0.

- Thus, the pFET network is the dual of the nFET network.

# Complex Logic Gates in CMOS

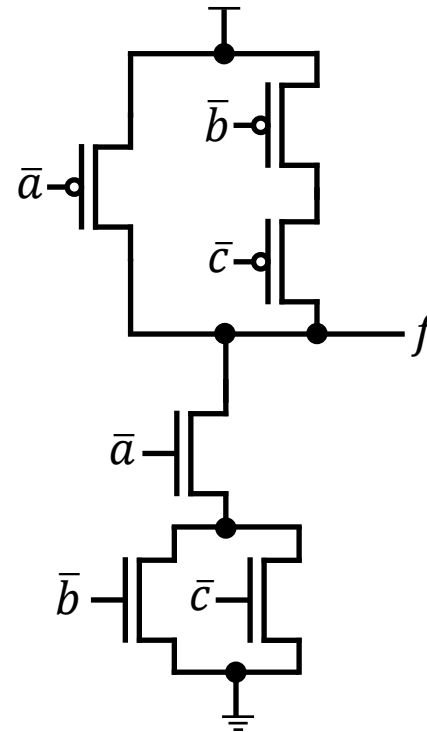
- Design methodology 3
  - When  $f = S(x_1, \dots, x_n)$  ( $S$  is a function of non-inverted variables)
    - $f = S = \bar{\bar{S}}$
    - Design  $\bar{S}$  and add an inverter at the output.
  - Example:  $f = a \cdot (b + c)$ 
    - $f = a \cdot (b + c) = \overline{\overline{a \cdot (b + c)}}$
    - Design  $\overline{a \cdot (b + c)}$ .
    - Add an inverter at the output.





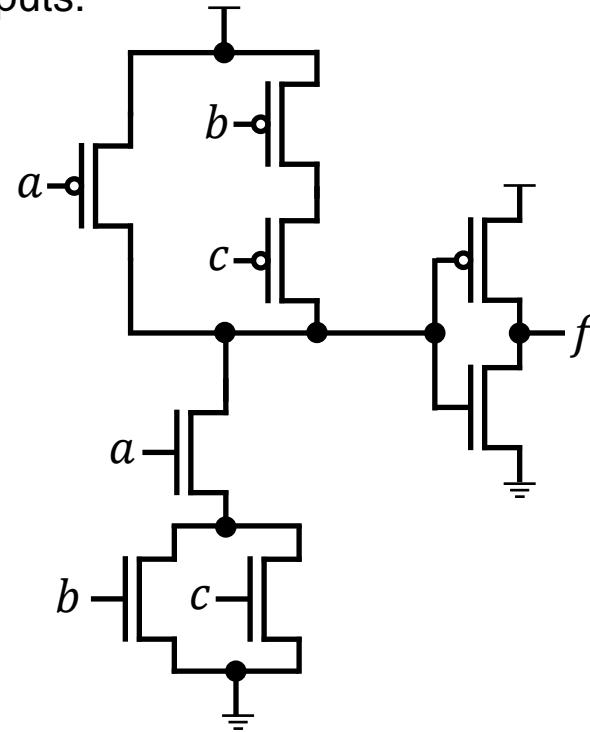
# Complex Logic Gates in CMOS

- Design methodology 4
  - When  $f = \overline{S(\overline{x_1}, \dots, \overline{x_n})}$  ( $S$  is a function of inverted variables)
    - Generate inverted inputs ( $\overline{x_1}, \dots, \overline{x_n}$ ) from the given inputs ( $x_1, \dots, x_n$ ).
    - Design  $S$  using the inverted inputs.
  - Example:  $f = \overline{\overline{a} \cdot (\overline{b} + \overline{c})}$ 
    - Inverters are not shown for brevity.



# Complex Logic Gates in CMOS

- Design methodology 5
  - When  $f = \overline{S(\overline{x_1}, \dots, \overline{x_n})}$ 
    - $f = \overline{S(\overline{x_1}, \dots, \overline{x_n})} = S(x_1, \dots, x_n)^D = \overline{\overline{S(x_1, \dots, x_n)^D}}$
    - Design  $\overline{S(x_1, \dots, x_n)^D}$  using the given inputs.
    - Add an inverter at the output.
  - Example:  $f = \overline{\overline{a} + (\overline{b} \cdot \overline{c})}$ 
    - $f = a \cdot (b + c) = \overline{\overline{a \cdot (b + c)}}$



# Complex Logic Gates in CMOS

- Design methodology 6
  - When  $f = S(\overline{x_1}, \dots, \overline{x_n})$ 
    - $f = S = \overline{\overline{S}}$
    - Generate inverted inputs  $(\overline{x_1}, \dots, \overline{x_n})$  from the given inputs  $(x_1, \dots, x_n)$ .
    - Design  $\overline{S}$  using the inverted inputs and add an inverter at the output.
- Design methodology 7
  - When  $f = S(\overline{x_1}, \dots, \overline{x_n})$ 
    - $f = \overline{\overline{S(\overline{x_1}, \dots, \overline{x_n})}} = \overline{S(x_1, \dots, x_n)^D}$
    - Design  $S^D$  using the given non-inverted inputs  $(x_1, \dots, x_n)$ .
- Design methodology 8
  - When  $f = S(x_1, \overline{x_1}, \dots, \overline{x_n})$  or  $\overline{\overline{S(x_1, \overline{x_1}, \dots, \overline{x_n})}}$ 
    - Convert the given function into an appropriate form to simplify the logic.
    - Design it.

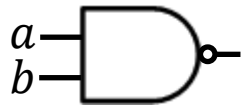
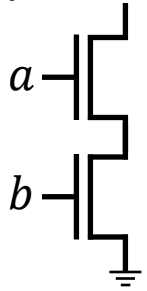
# Complex Logic Gates in CMOS

- Examples (assuming only non-inverted inputs are available)
  - $f = a \cdot b$  (AND2)
    - Design  $f = \overline{a \cdot b}$  and add an inverter at the output. (# TRs: 6)
    - Design  $f = \overline{\overline{a \cdot b}} = \overline{\overline{a} + \overline{b}}$  with two inverters to generate  $\overline{a}$  and  $\overline{b}$ . (# TRs: 8)
  - $f = \overline{\overline{a \cdot b + c \cdot d}}$ 
    - Add two inverters to generate  $\overline{a}$  and  $\overline{c}$ , then design  $f$ . (# TRs: 12)
  - $f = \overline{s} \cdot a + s \cdot b$  (2:1 MUX)

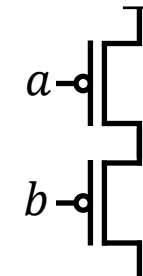
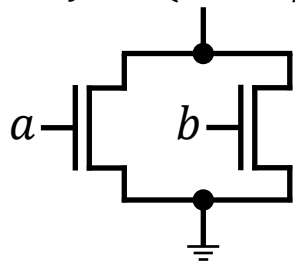
# Complex Logic Gates in CMOS

- Bubble pushing (how to construct a pFET network)
  - $f = A \cdot 1 + B \cdot 0 = F(\bar{x}_1, \dots, \bar{x}_n) \cdot 1 + F(x_1, \dots, x_n)^D \cdot 0$

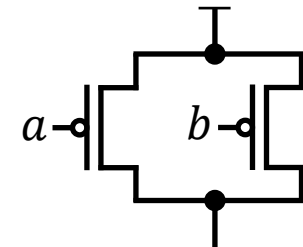
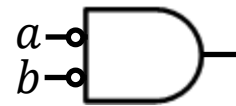
$$f = a \cdot b \cdot 0$$



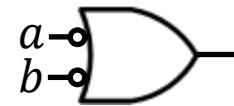
$$f = (a + b) \cdot 0$$



$$f = \bar{a} \cdot \bar{b} \cdot 1$$

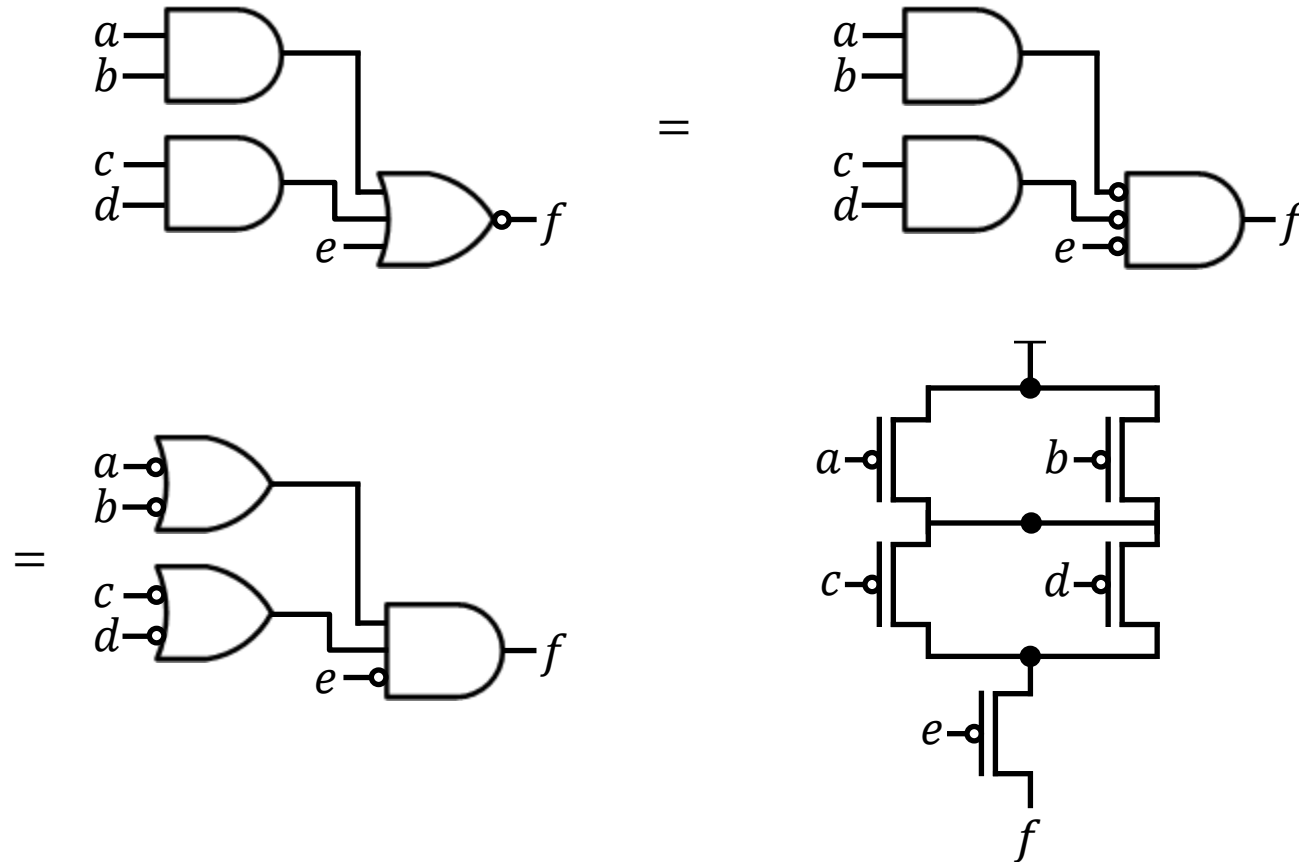


$$f = (\bar{a} + \bar{b}) \cdot 1$$



# Complex Logic Gates in CMOS

- Bubble pushing (how to construct a pFET network)
  - Example



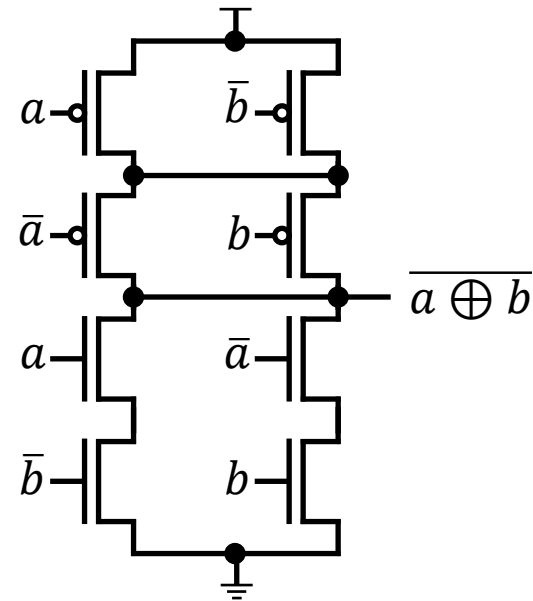
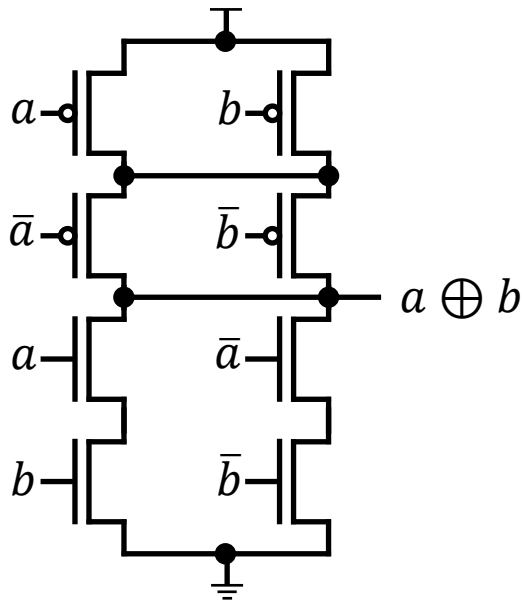
# Complex Logic Gates in CMOS

- XOR

- $a \oplus b = a \cdot \bar{b} + \bar{a} \cdot b = \overline{a \cdot b + \bar{a} \cdot \bar{b}}$  (#TRs: 8+4(for the two inverters))

- XNOR

- $\overline{a \oplus b} = a \cdot b + \bar{a} \cdot \bar{b} = \overline{a \cdot \bar{b} + \bar{a} \cdot b}$  (#TRs: 8+4(for the two inverters))

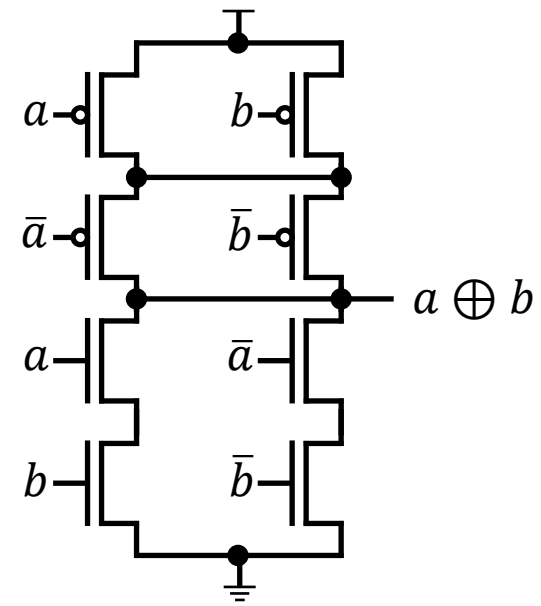


# Complex Logic Gates in CMOS

- Structured logic analysis
  - Derive a Boolean equation for a given transistor-level schematic.
- Analysis methodology 1
  - Convert the nFET network into a Boolean equation (only when the pFET network is the dual of the nFET network.)
  - Notice that the nFET network passes logic 0.

- Example

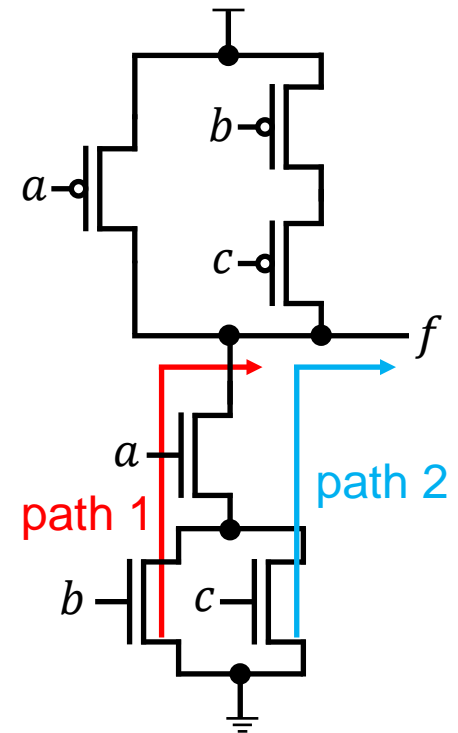
- $f = \overline{a \cdot b + \bar{a} \cdot \bar{b}} = (\bar{a} + \bar{b}) \cdot (a + b) = a \cdot \bar{b} + \bar{a} \cdot b$





# Complex Logic Gates in CMOS

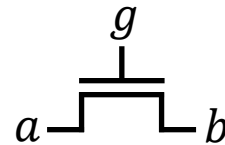
- Analysis methodology 2
  - Identify all the paths from  $V_{SS}$  to the output (only when the pFET network is the dual of the nFET network.)
  - Merge them into a single Boolean equation.
  - Negate the output.
- Example
  - Path 1:  $b \cdot a$
  - Path 2:  $c \cdot a$
  - Merge:  $b \cdot a + c \cdot a = a \cdot (b + c)$
  - Negate:  $\overline{a \cdot (b + c)}$
  - $f = \overline{a \cdot (b + c)}$



# Pass Transistors

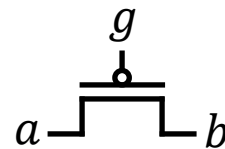
- nFET

- $g = 0$ : OFF
- $g = 1$ : ON
  - $a = 0$ :  $b = \text{strong } 0$
  - $a = 1$ :  $b = \text{weak } 1$



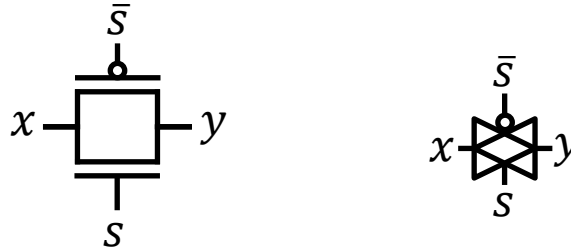
- pFET

- $g = 1$ : OFF
- $g = 0$ : ON
  - $a = 0$ :  $b = \text{weak } 0$
  - $a = 1$ :  $b = \text{strong } 1$



# Transmission Gate Circuits

- Transistor circuit



- Behaviors

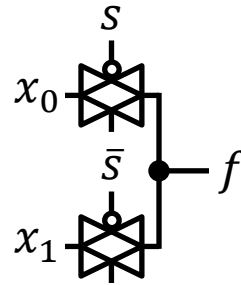
- When  $s = 0$ : Both nFET and pFET are OFF.
- When  $s = 1$ : Both nFET and pFET are ON.
  - If  $x = 0$ , the nFET perfectly transmits it to  $y$  (nFET is good at transferring 0.)
  - If  $x = 1$ , the pFET perfectly transmits it to  $y$  (pFET is good at transferring 1.)

- Disadvantage

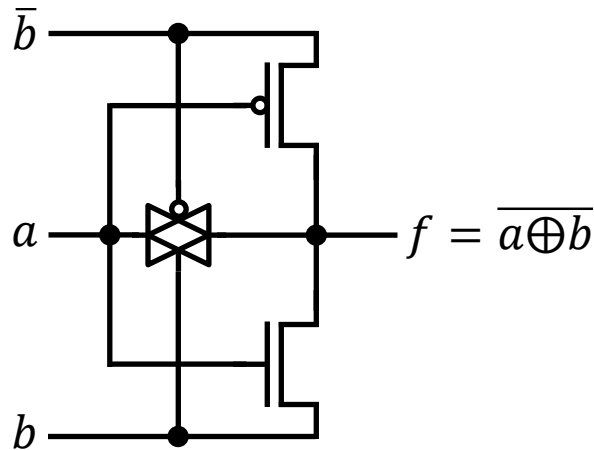
- Needs  $\bar{s}$ .
- Does not restore the input signals.

# Transmission Gate Circuits




- Logic design using transmission gates
  - MUX:  $f = \bar{s} \cdot x_0 + s \cdot x_1$



- XNOR

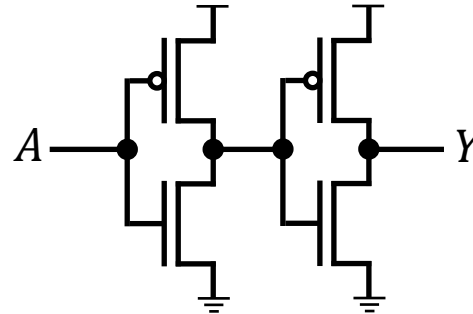


# Pass Transistors vs. Transmission Gates

	Pass TR.		Transmission Gates
Symbols			
Signal strength	Strong 0 Weak 1	Weak 0 Strong 1	Strong 0 Strong 1
Area	$A$	$rA (r > 1)$	$(1 + r)A$
Control signal	$g$	$g$	$g, \bar{g}$

# Buffer

- $Y = A$



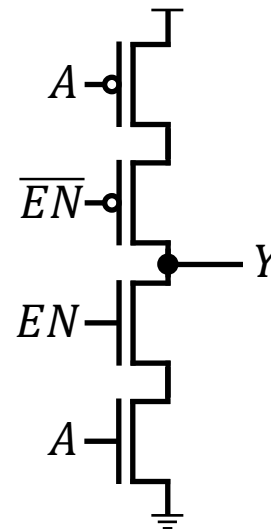
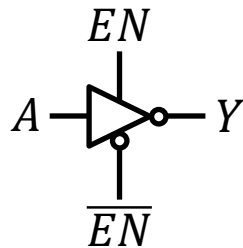
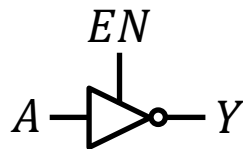
- Buffers are used for
  - Signal restoration
  - Interconnect optimization

# Tristate Inverter

- Truth table

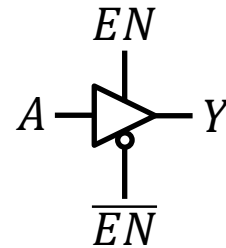
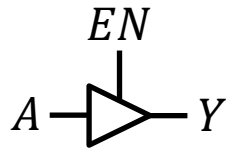
EN	Y
0	Z
1	$\bar{A}$

- Symbol & Schematic

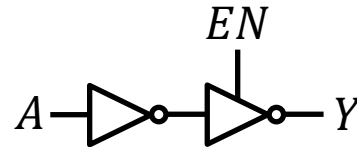


# Tristate Buffer

- Symbol



- Gate-level schematic

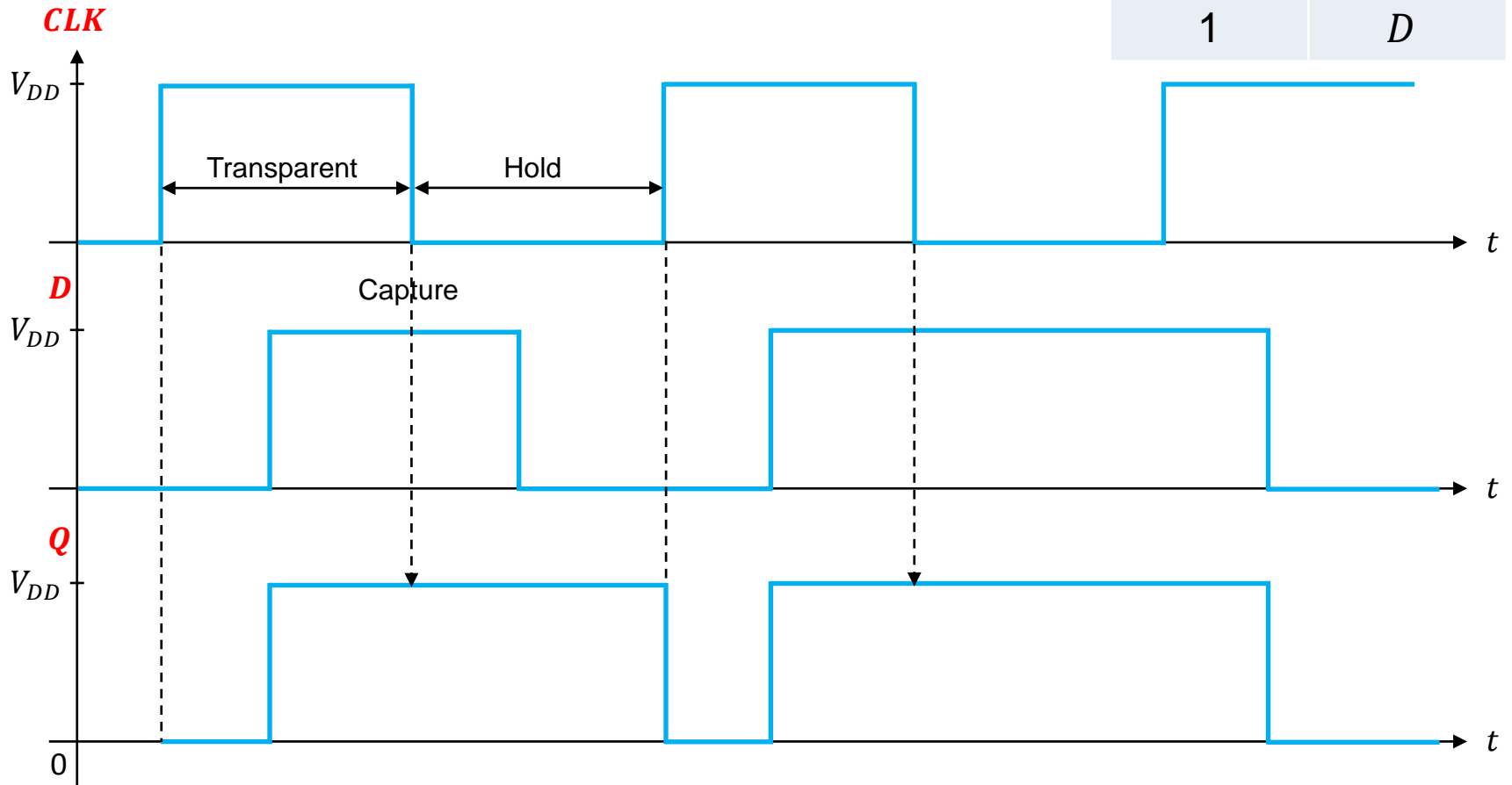




# Sequential Circuit – D Latch

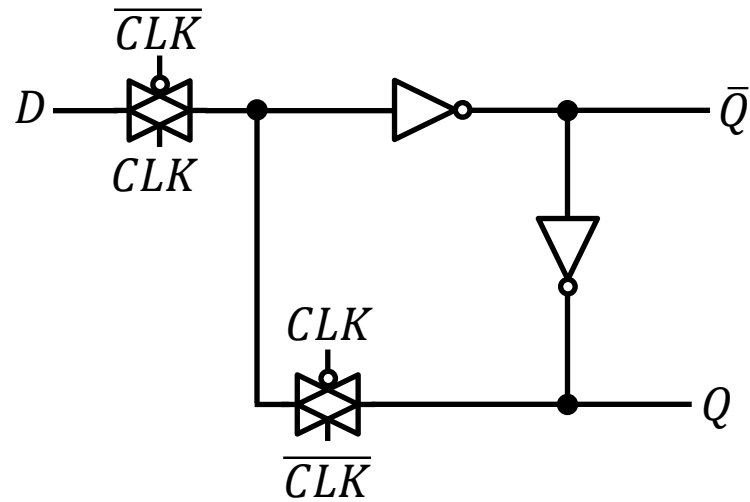
- Positive-level-sensitive D latch

CLK	Q
0	hold
1	D



# Sequential Circuit – D Latch

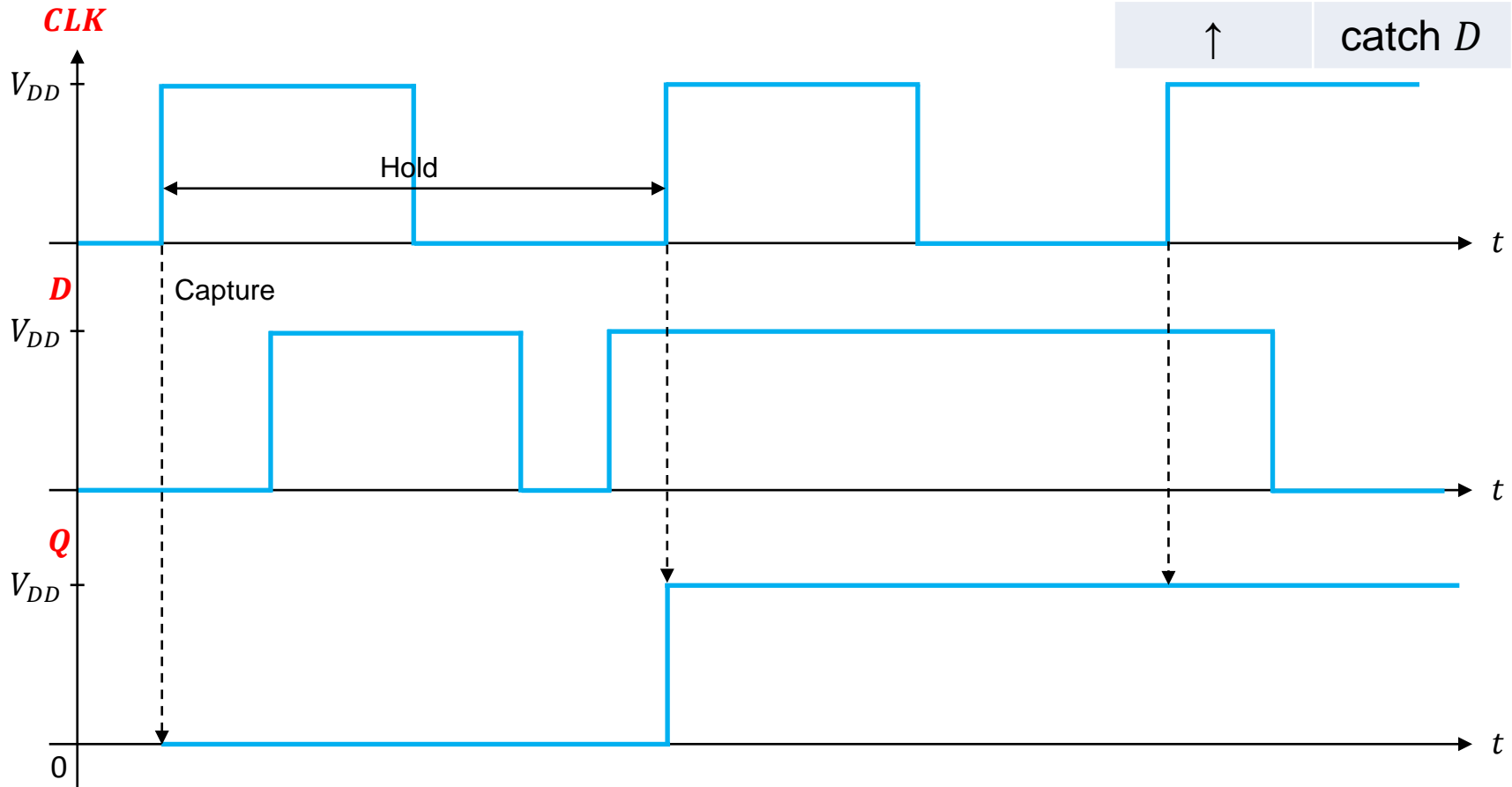
- Schematic



# Sequential Circuit – D Flip-Flop

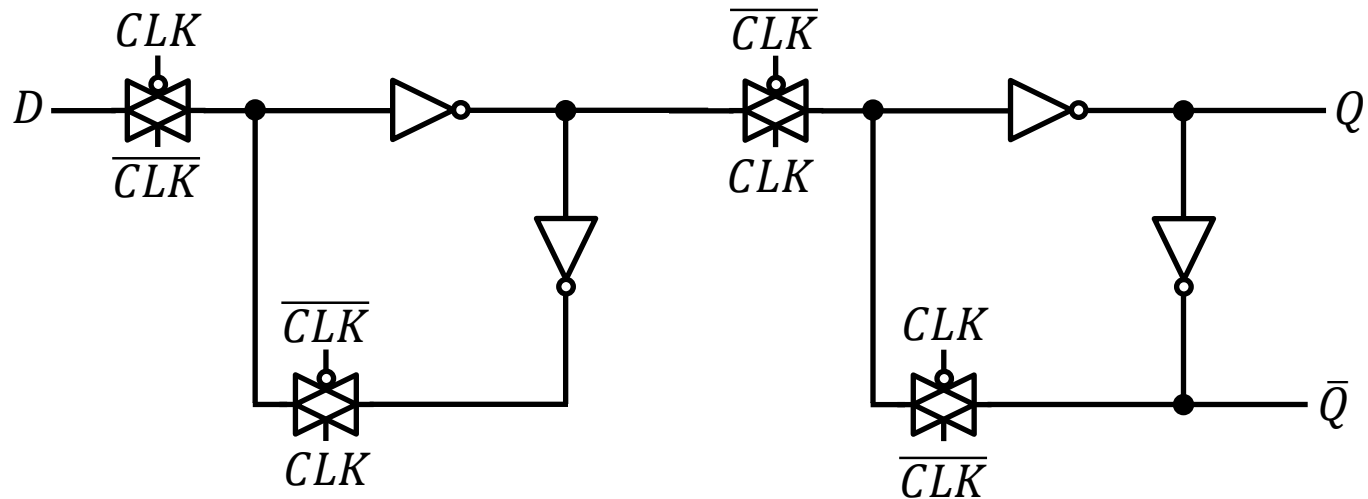
- Positive-edge-triggered D flip-flop

CLK	Q
0, 1	hold
↑	catch $D$



# Sequential Circuit – D Flip-Flop

- Schematic



# Sequential Circuit

- Example
  - Inputs:  $D, ARN, CLK, \overline{CLK}$
  - Outputs:  $Q, \overline{Q}$

