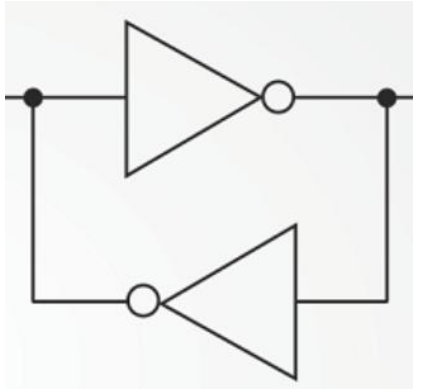
Random Access Memory (RAM)

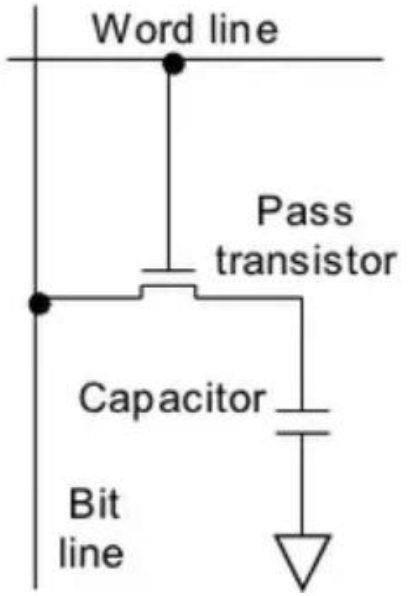
=

Two types:

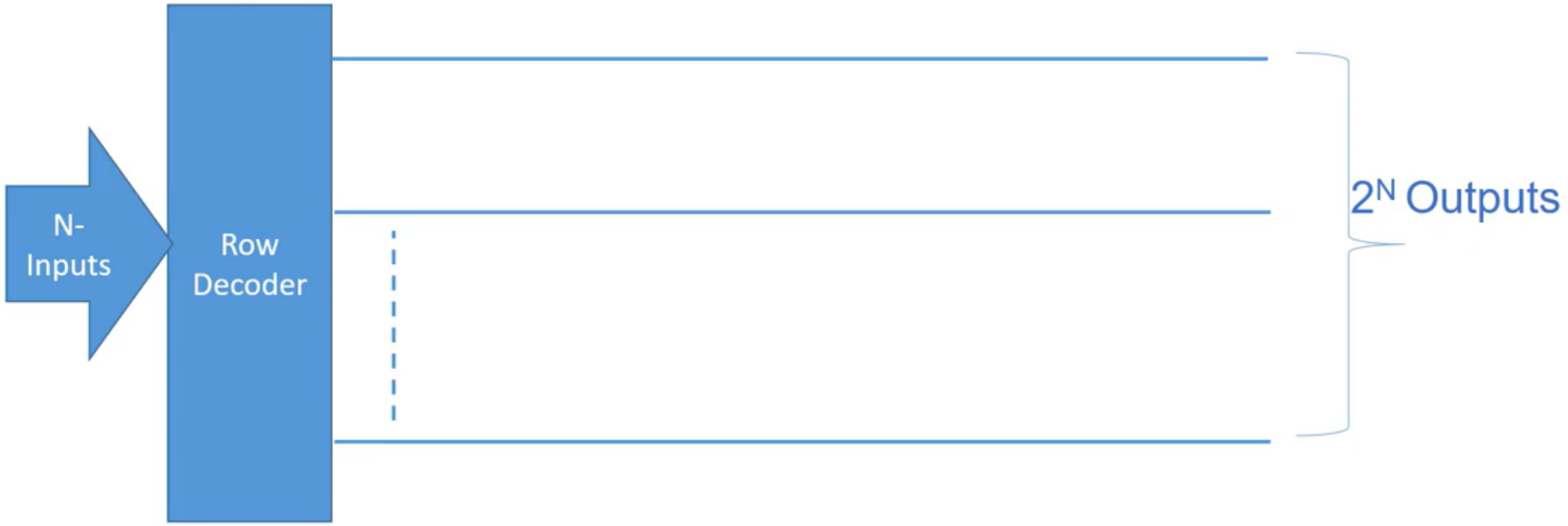
1. **Static RAM (SRAM):** Stored information depends on the state of transistor, and it requires continuous power supply.



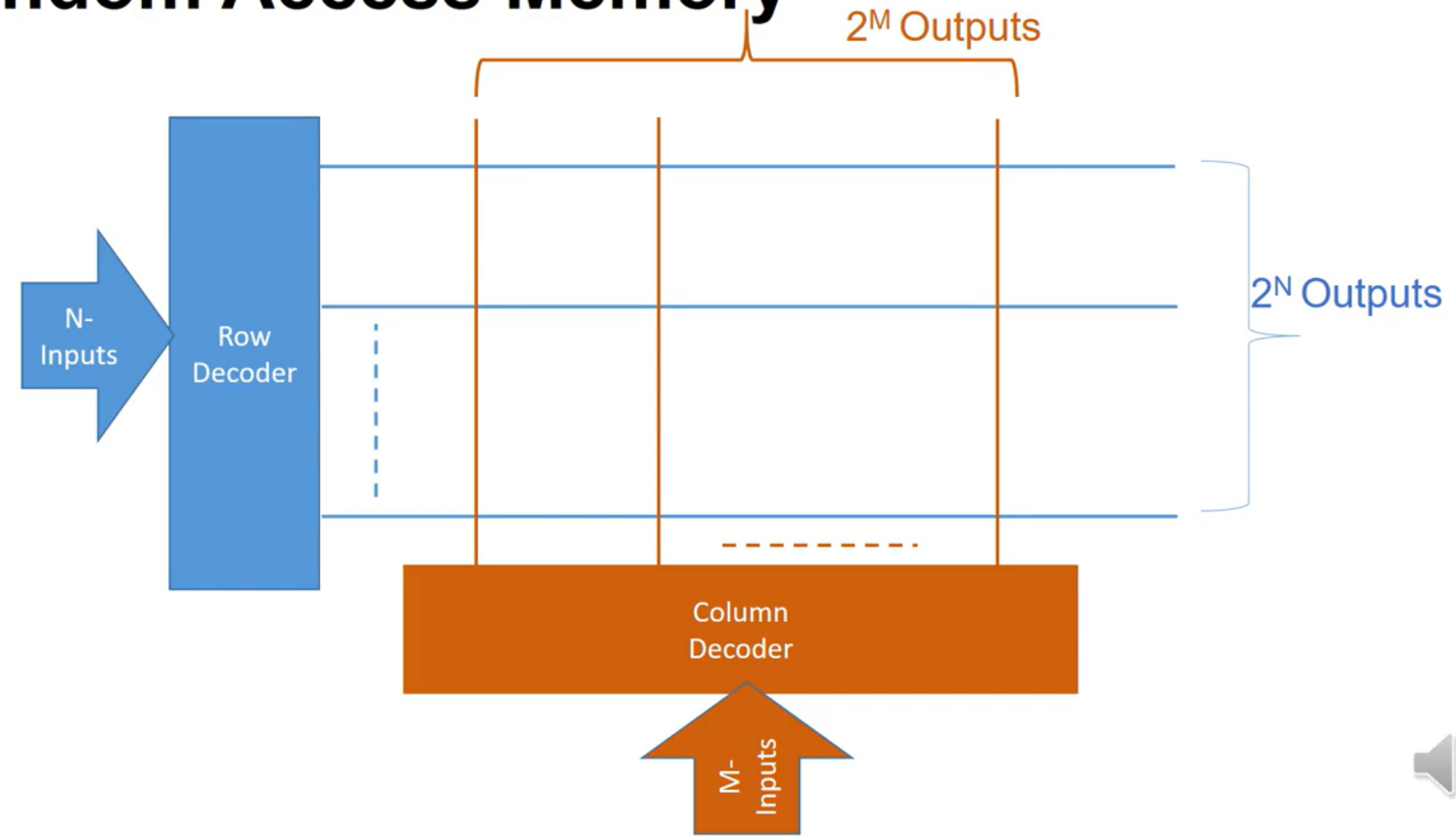
2. **Dynamic RAM (DRAM):** Stored information depends on charge in capacitor and doesn't require continuous power supply but the charge in the capacitor must be refreshed after a particular interval of time.



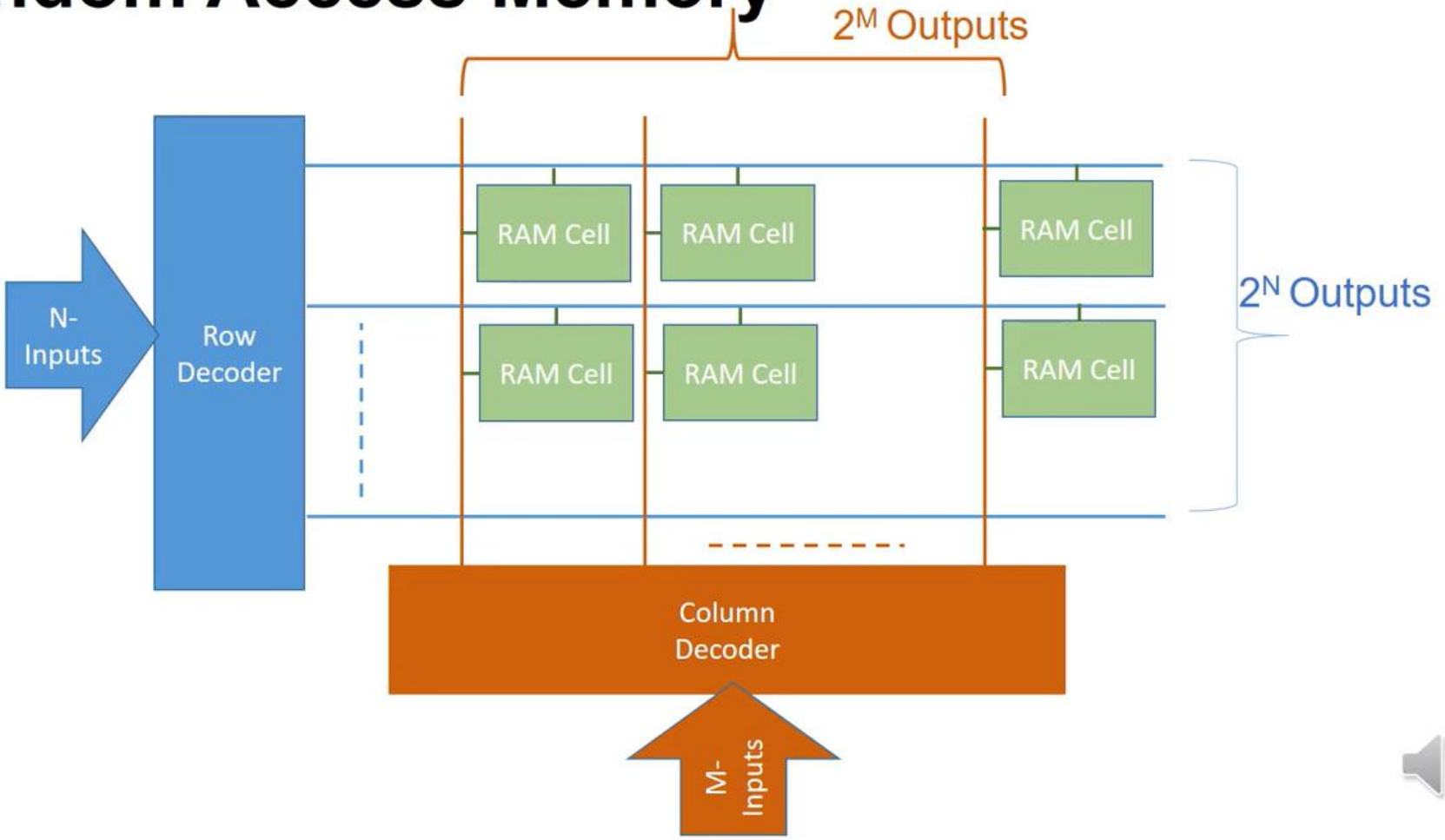
Random Access Memory



Random Access Memory

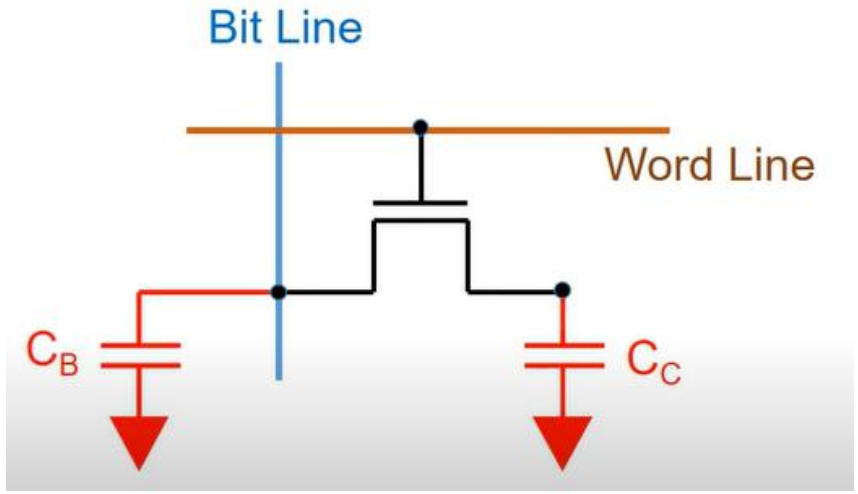


Random Access Memory

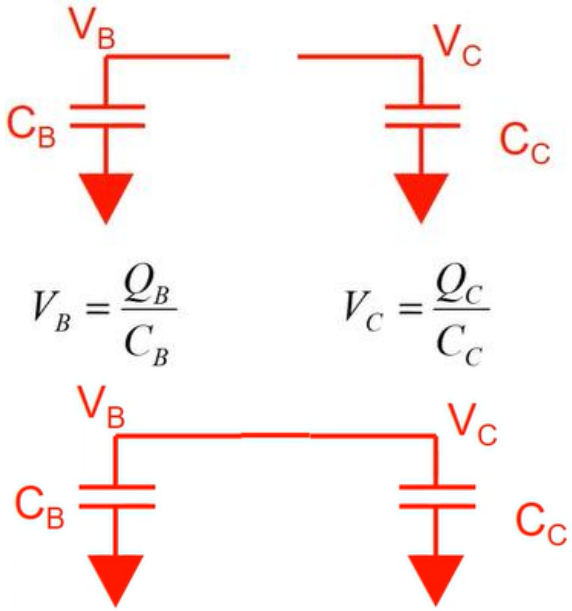


1 transistor- Dynamic RAM (1T-DRAM)

1 transistor- Dynamic RAM (1T-DRAM)



- **Word line:** It is the output of row decoder. If you want to read or write data then word line should be **high.**
- **Bit line:** It is the output of column decoder. To write any data, the data is supplied through bit line. To read any data, the data is read through bit line.



$$V_B = \frac{Q_B}{C_B} \qquad V_C = \frac{Q_C}{C_C}$$

Final voltage $$V_F = \frac{Q_B + Q_C}{C_B + C_C}$$

Difference in bitline voltage

$$\Delta V_B = V_B - V_F$$

$$\Delta V_B = \frac{Q_B}{C_B} - \frac{Q_B + Q_C}{C_B + C_C}$$

$$\Delta V_B = \frac{Q_B(C_B + C_C) - C_B(Q_B + Q_C)}{C_B(C_B + C_C)}$$

$$\Delta V_B = \frac{Q_B\left(1 + \frac{C_C}{C_B}\right) - (Q_B + Q_C)}{(C_B + C_C)}$$

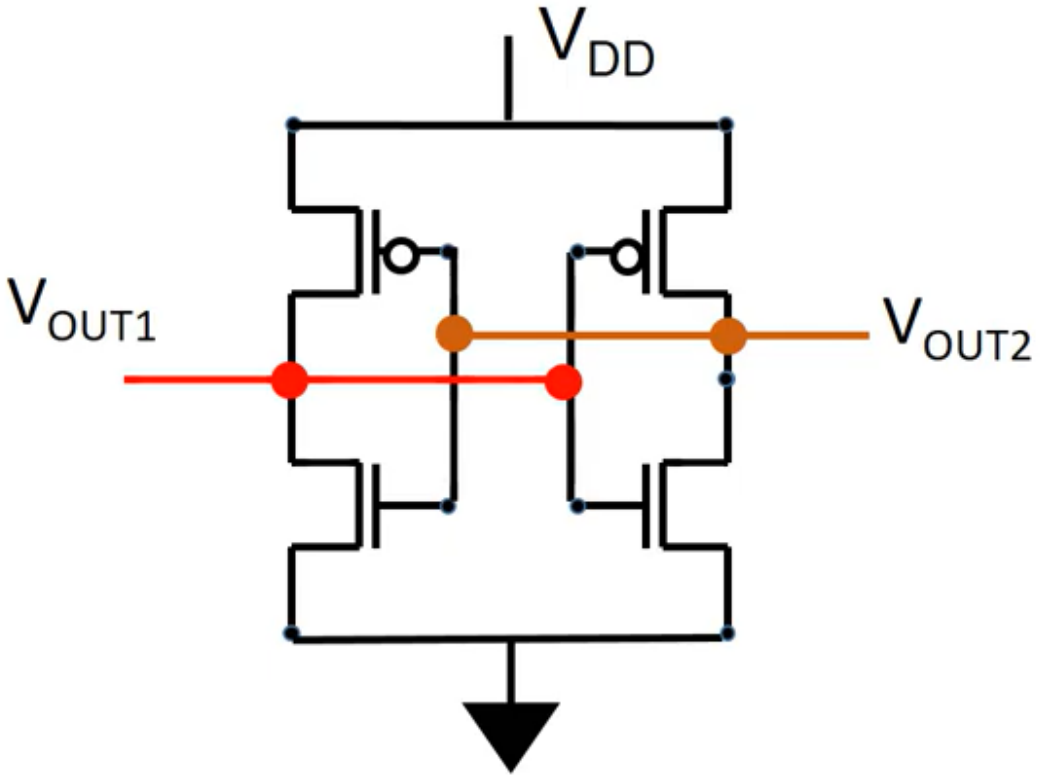
$$\Delta V_B = \frac{C_C(V_B - V_C)}{(C_B + C_C)}$$

Assume V_B=2.5V, V_C=5V, C_B=1pF, and C_C=50fF,

$$\Delta V_B = -119mV$$

V_F = 2.381 V or 2.619 V. Hence, the data is destroyed

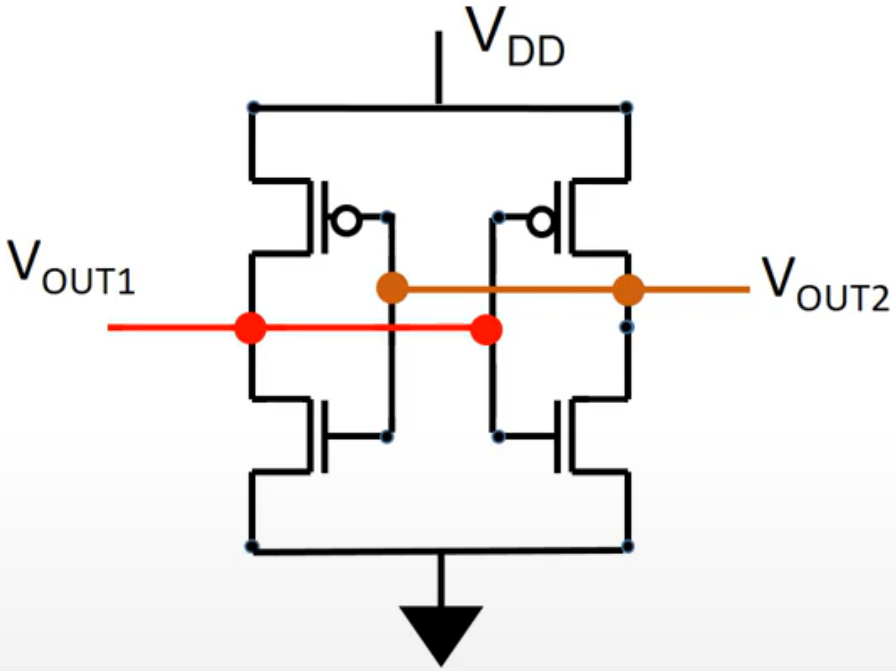
=



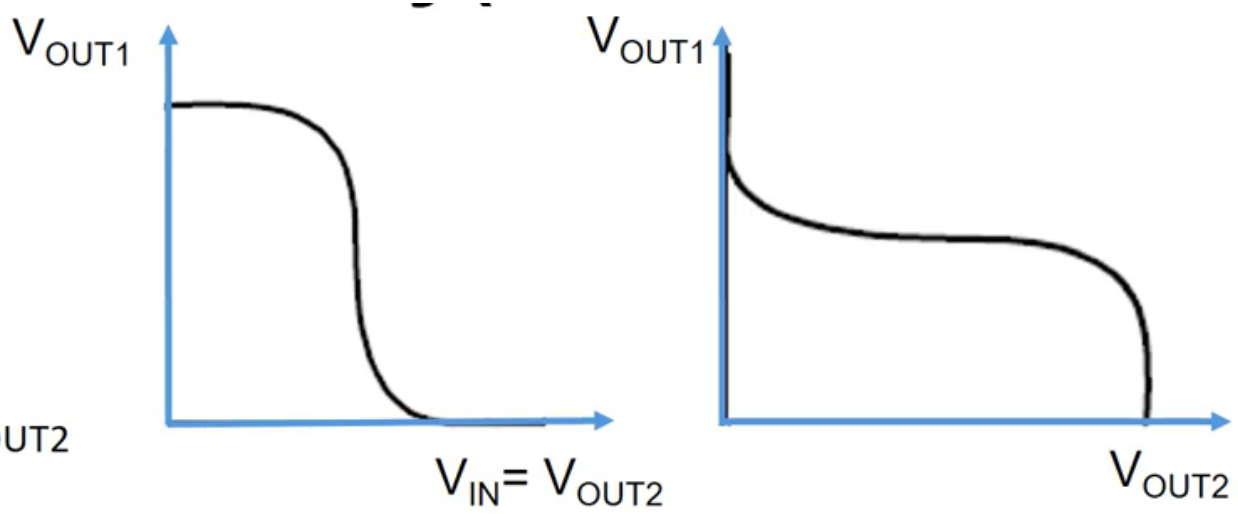
$V_{OUT1} = V_{OUT2} = 0.5V_{DD}$

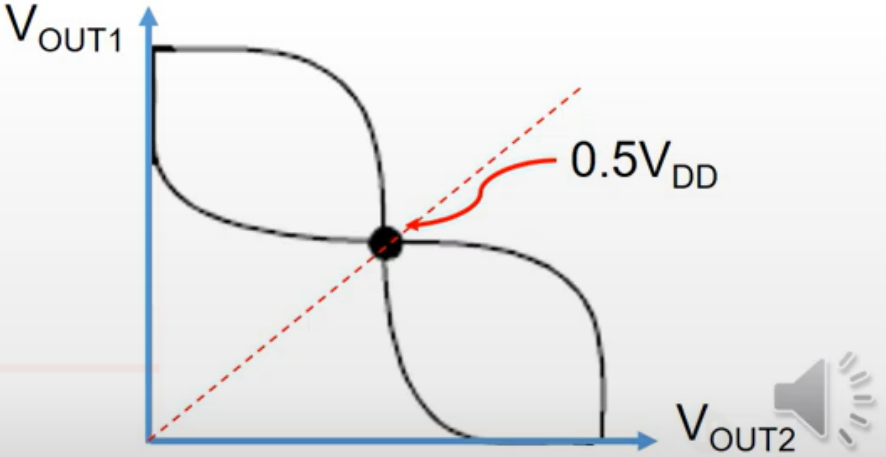
Sense Amplifier

Sense Amplifier

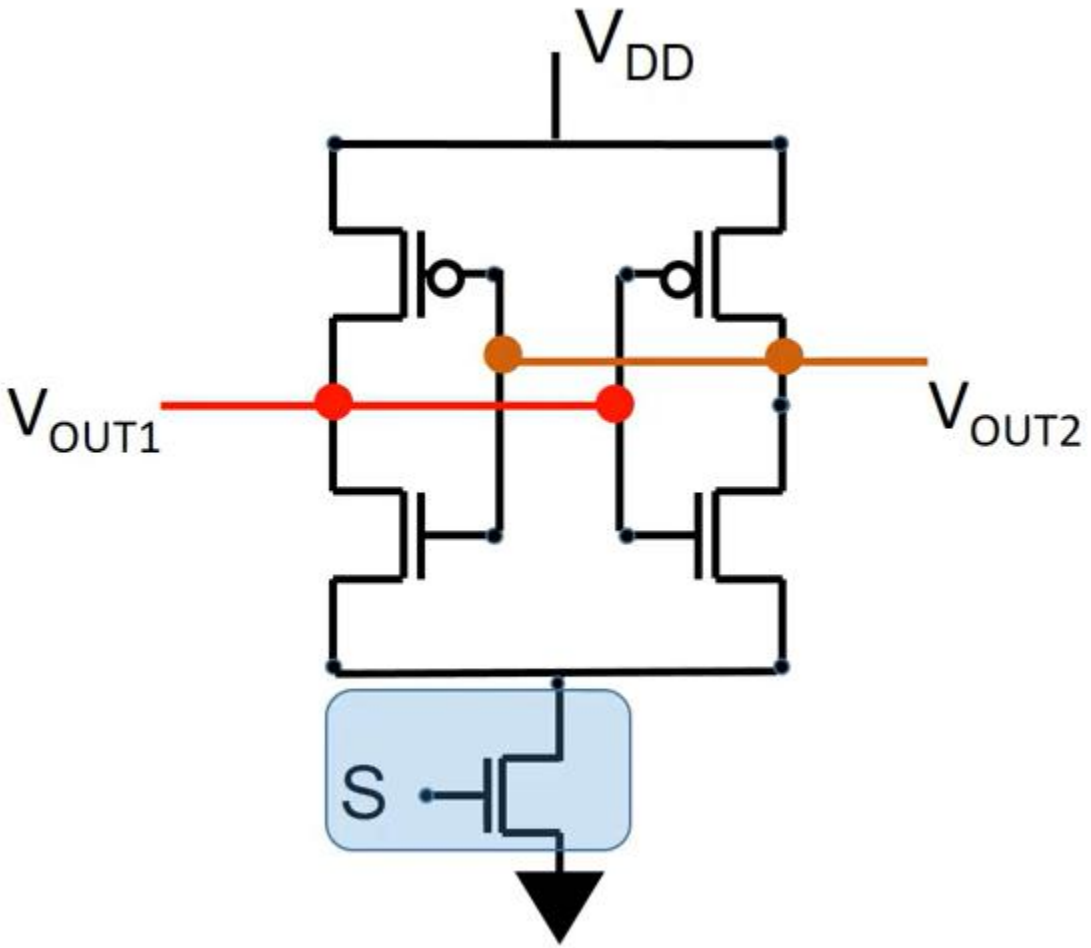


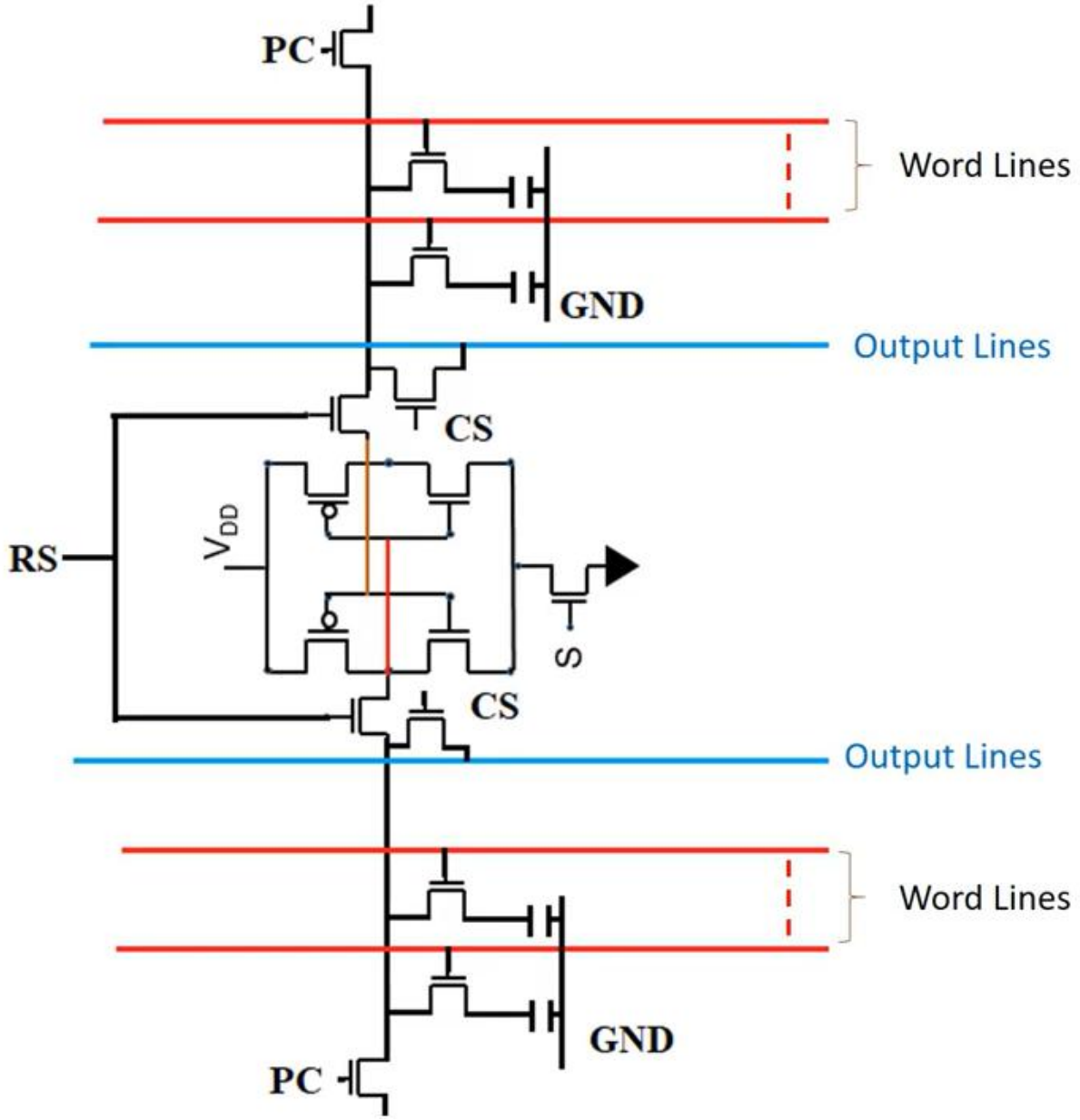
$V_{OUT1} = V_{OUT2} = 0.5V_{DD}$

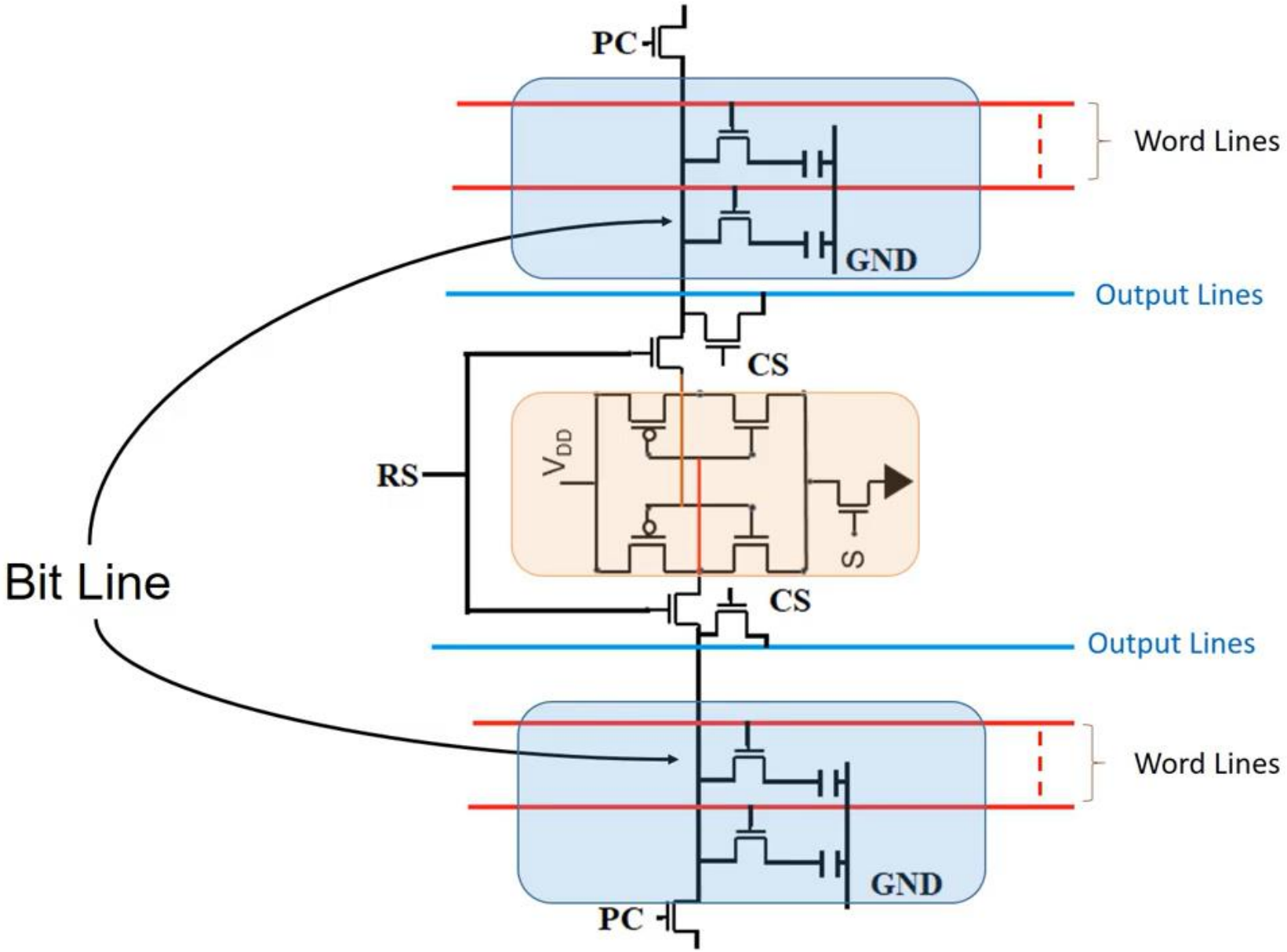


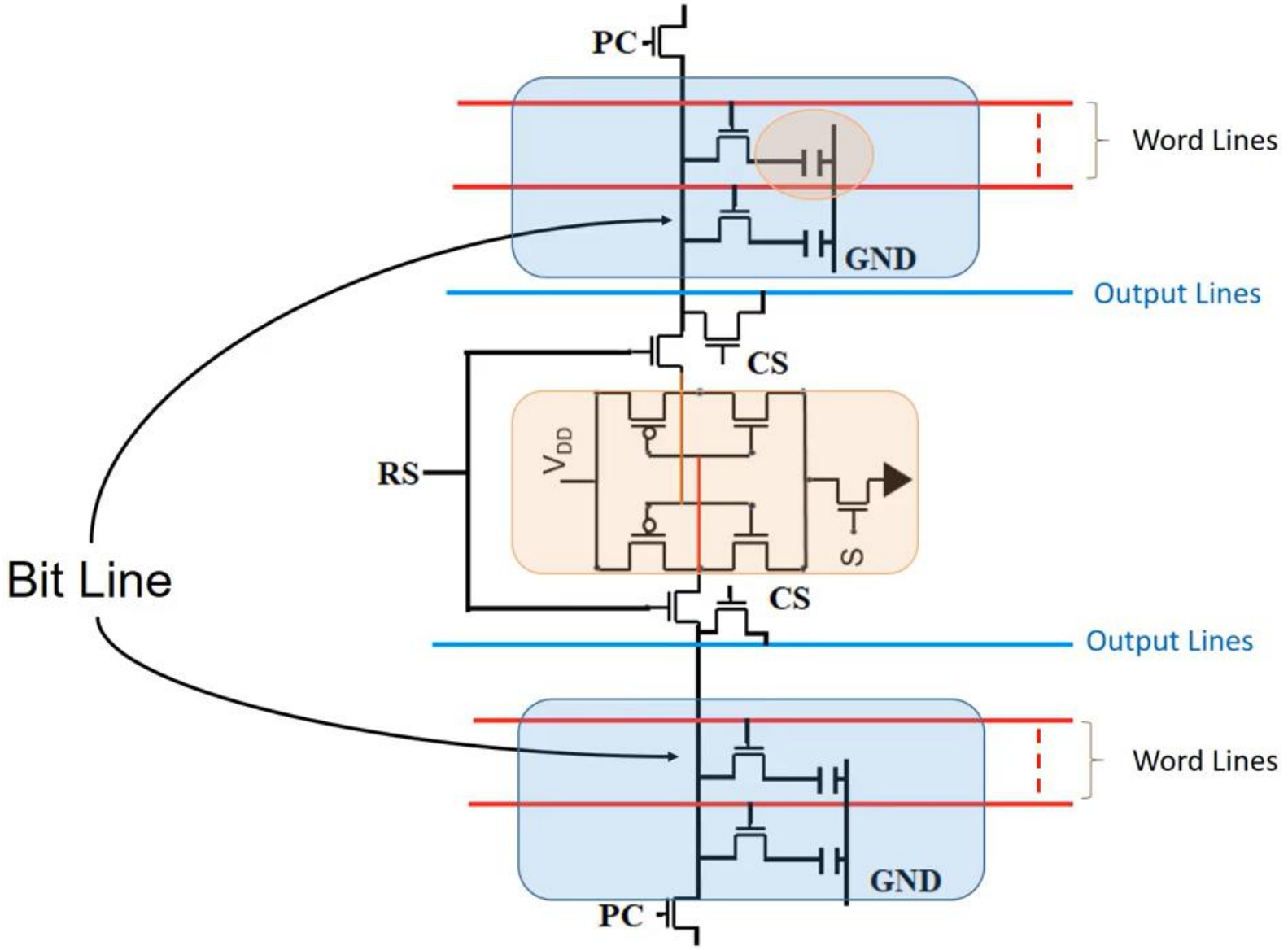


=


V_{DD}
V_{OUT1}
V_{OUT2}
S


PC
Word Lines
GND
Output Lines
CS
RS
V_{DD}
S
CS
Output Lines
Word Lines
GND
PC


PC
Word Lines
GND
Output Lines
CS
RS
V_{DD}
S
CS
Bit Line
Output Lines
Word Lines
GND
PC


PC
Word Lines
GND
Output Lines
CS
RS
V_{DD}
S
Bit Line
CS
Output Lines
Word Lines
GND
PC

Random Access Memory(1-T DRAM)

- **Write operation** : 1 T dynamic memory cell consists of a capacitor C_c which can be **charged** from **read/write line(Bit line)**, provided that the Word Line is high

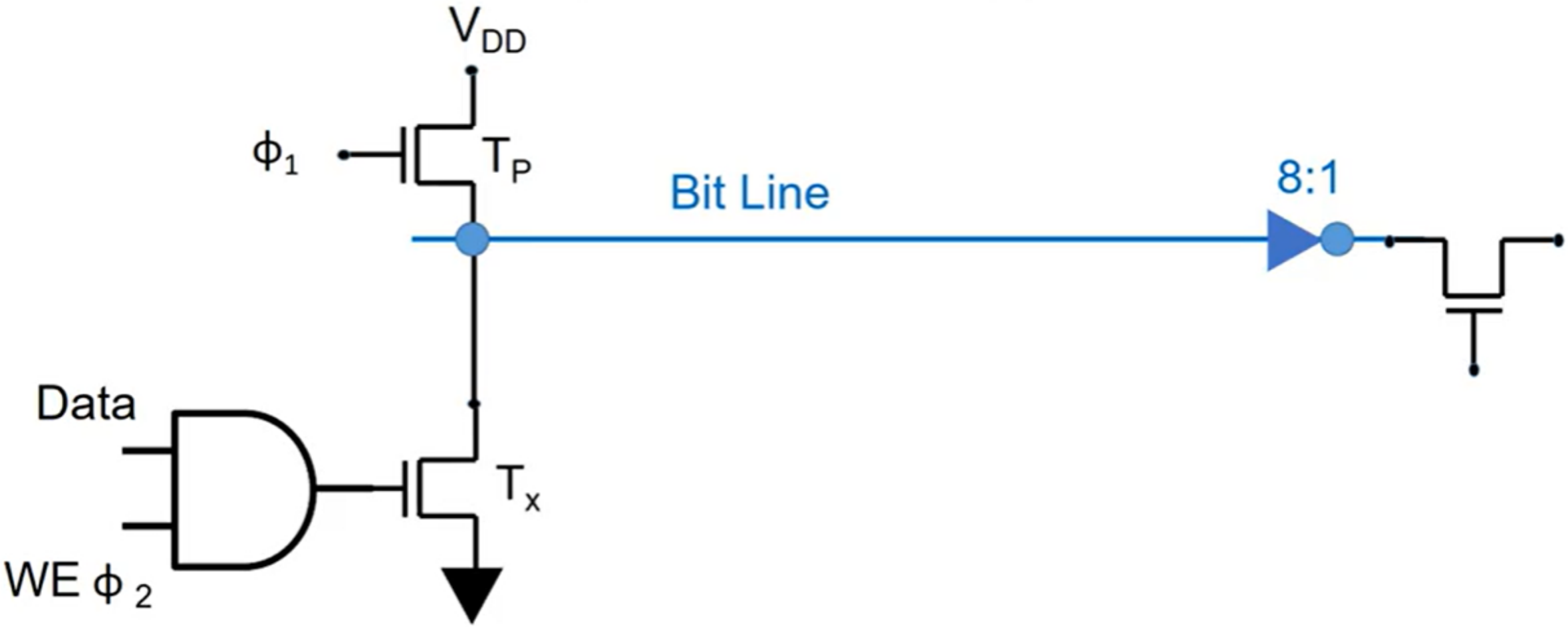
- **Read operation** : The state of the charge across C_c can be read via the same read/write line with a high Word Line . Hence, *during read operation, a sense amplifier is used to differentiate between a stored 0 and a stored 1.*

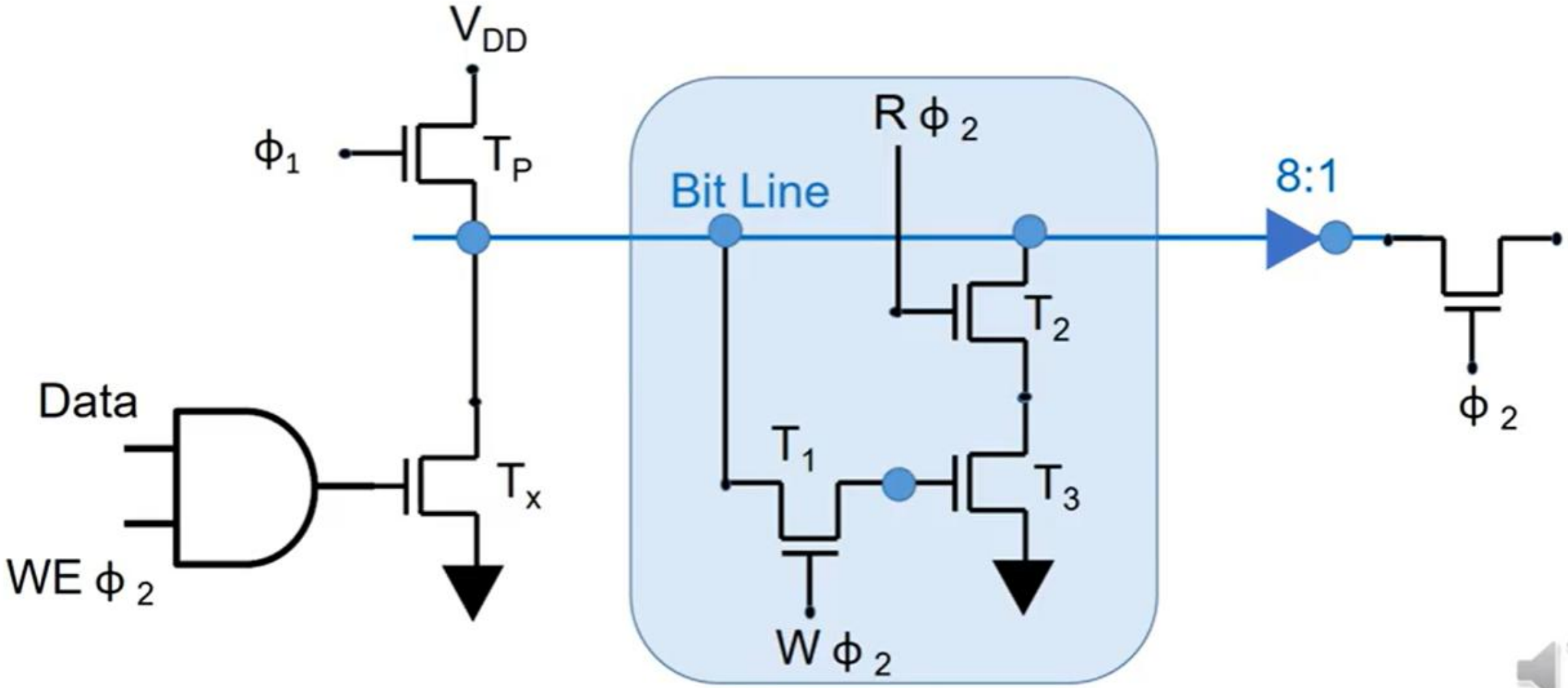
3T-DRAM

3T DRAM uses 2 Phase Clock Signal

- Two clocks cannot be high at the same time.
- Two clocks can be low at the same time.

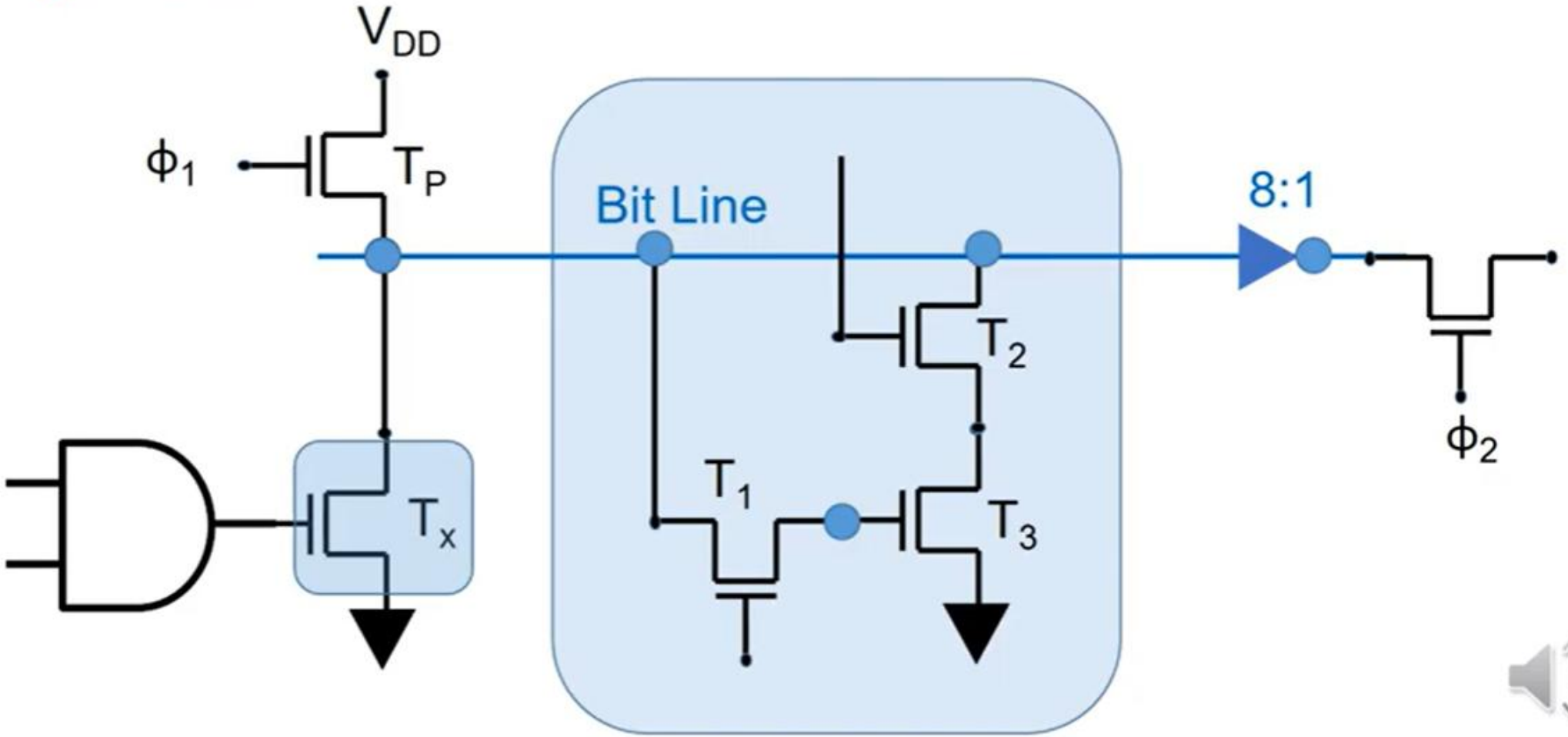
=


V_{DD}
ϕ_1
T_P
Bit Line
8:1
Data
T_x
WE ϕ_2


V_{DD}
ϕ_1
T_P
Bit Line
R ϕ_2
8:1
T_2
ϕ_2
Data
T_1
T_3
T_x
WE ϕ_2
W ϕ_2

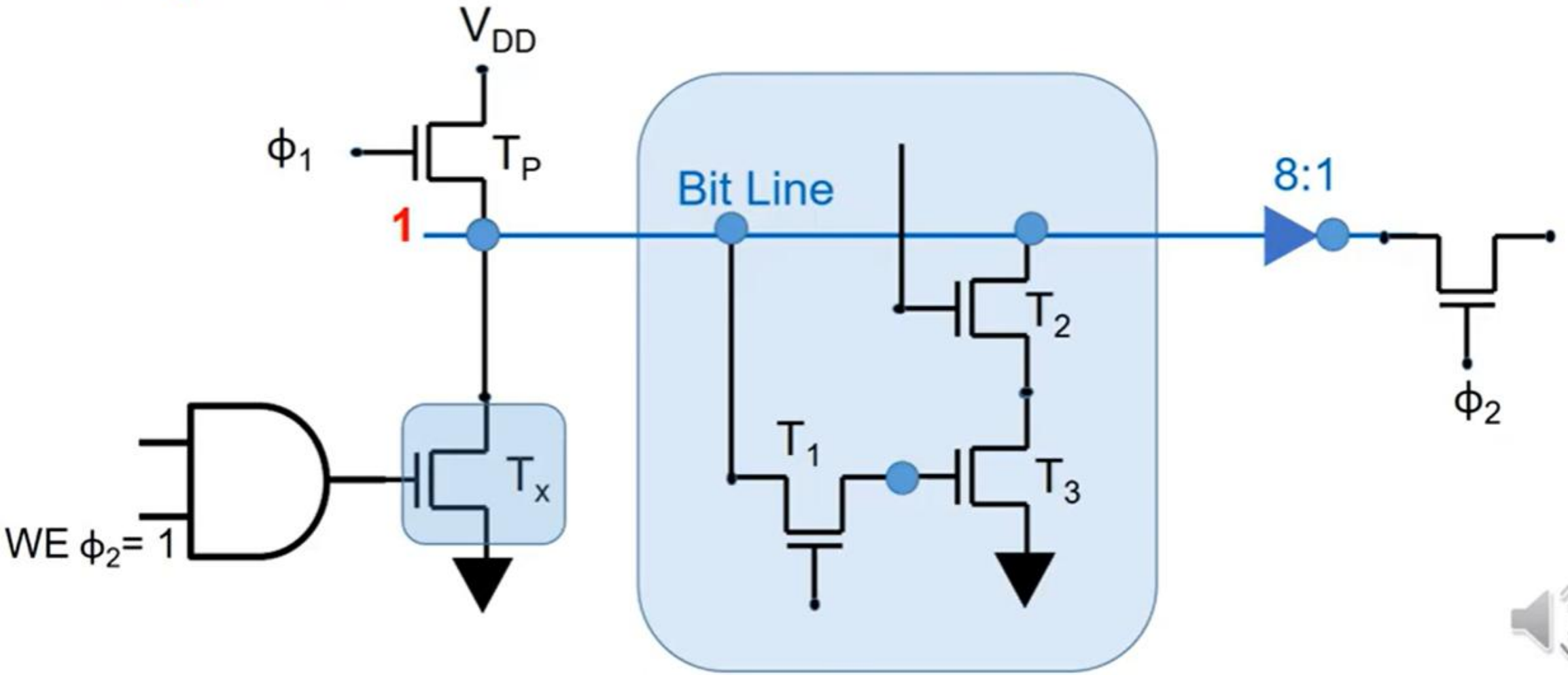
Write 1

Write Operation



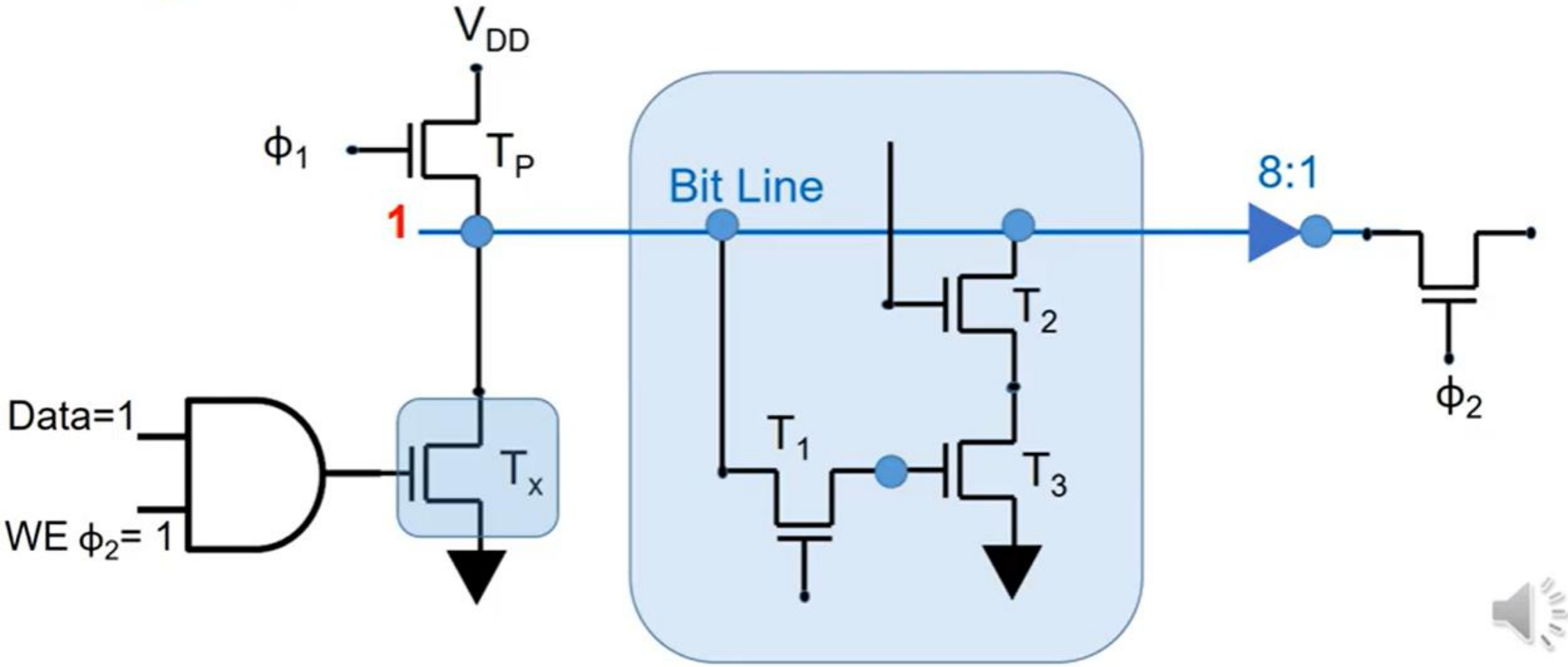
Write 1

Write Operation



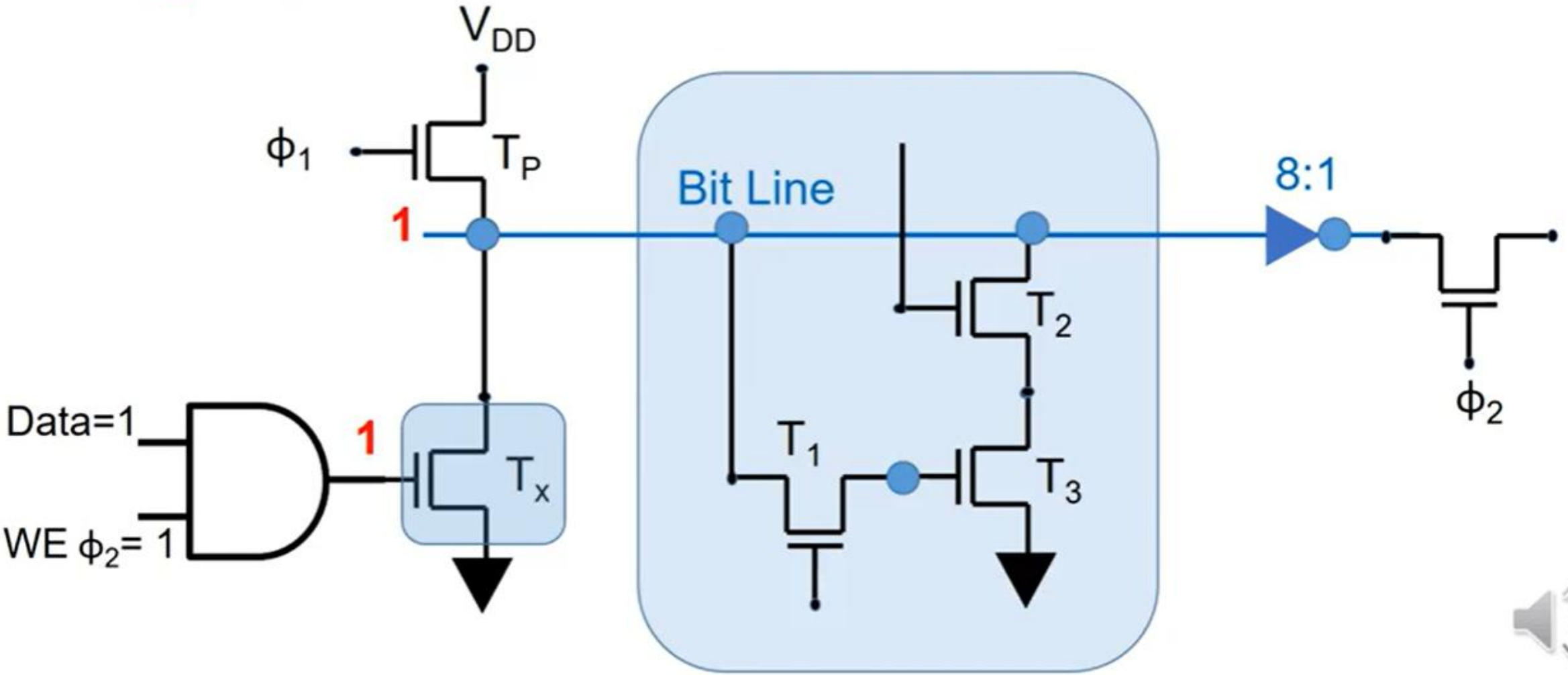
Write 1

Write Operation



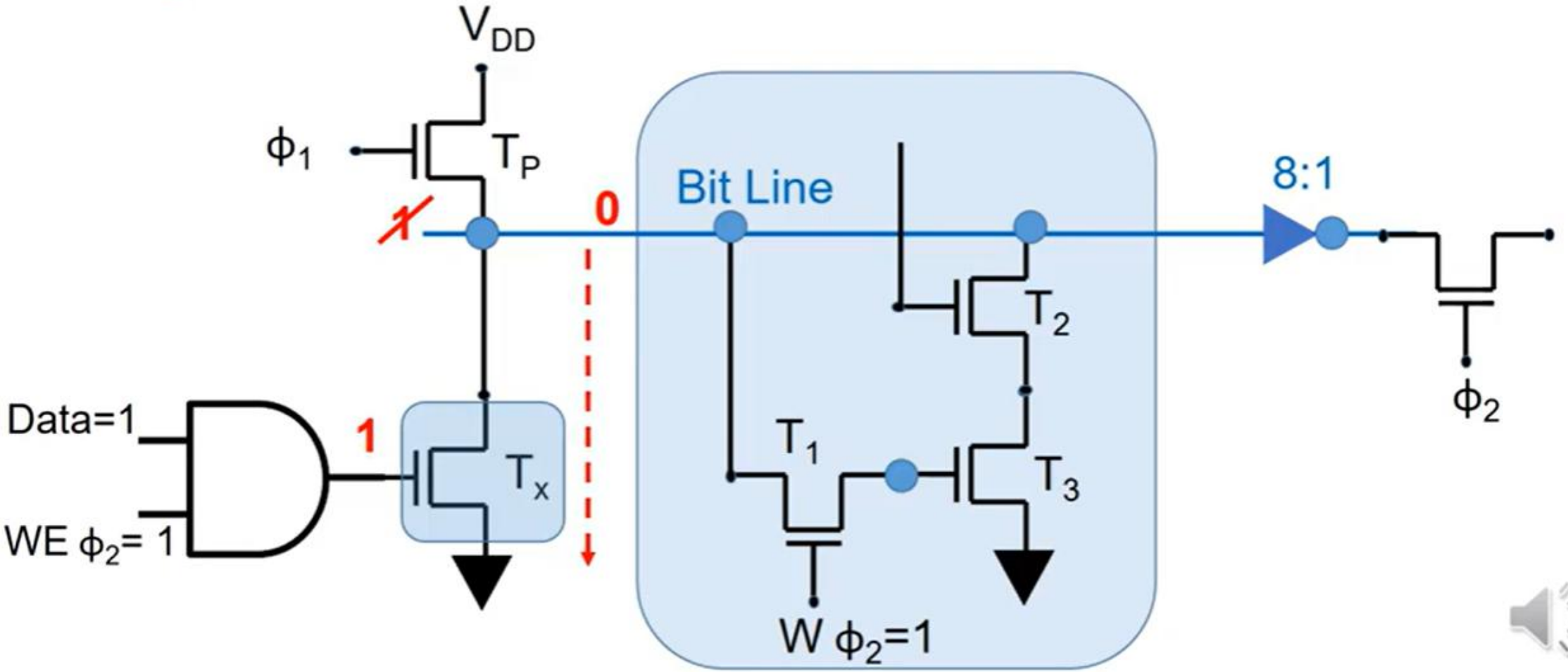
Write 1

Write Operation



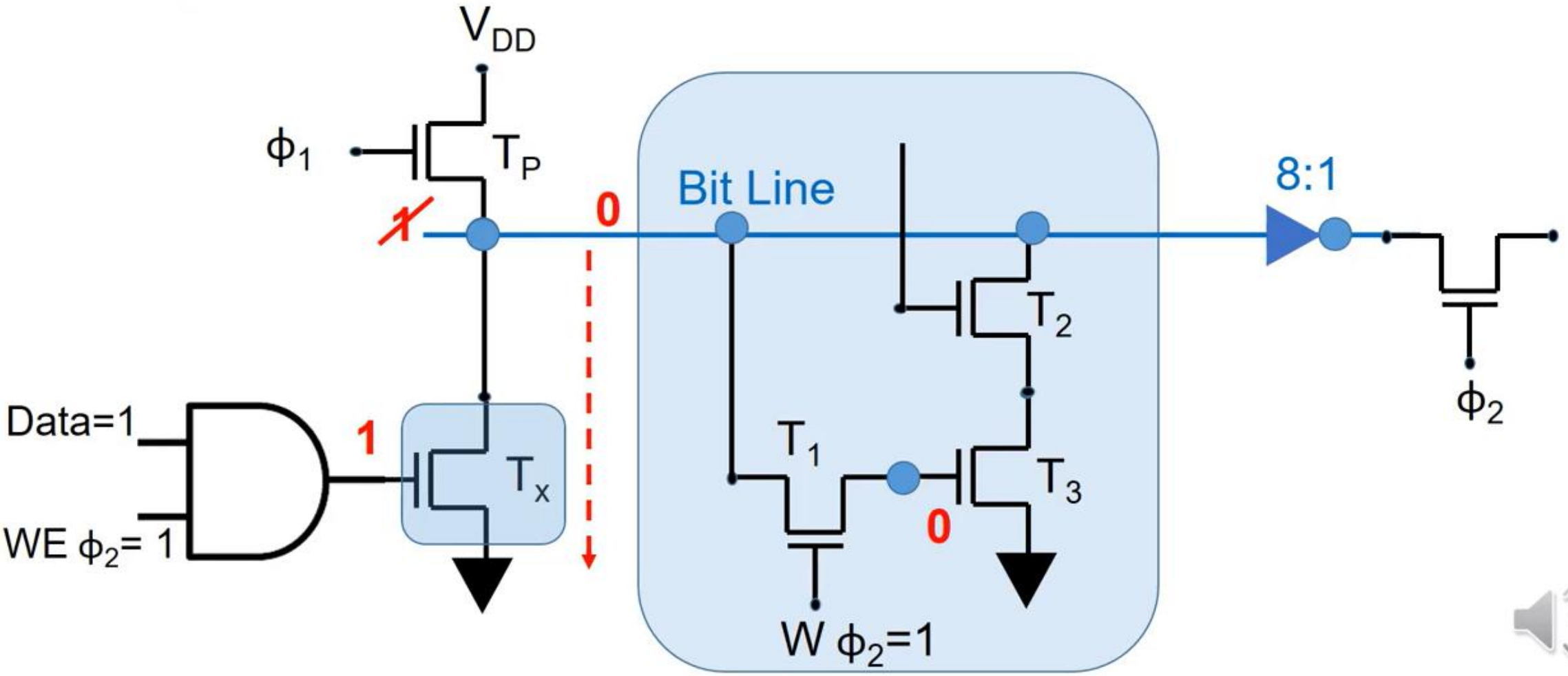
Write 1

Write Operation



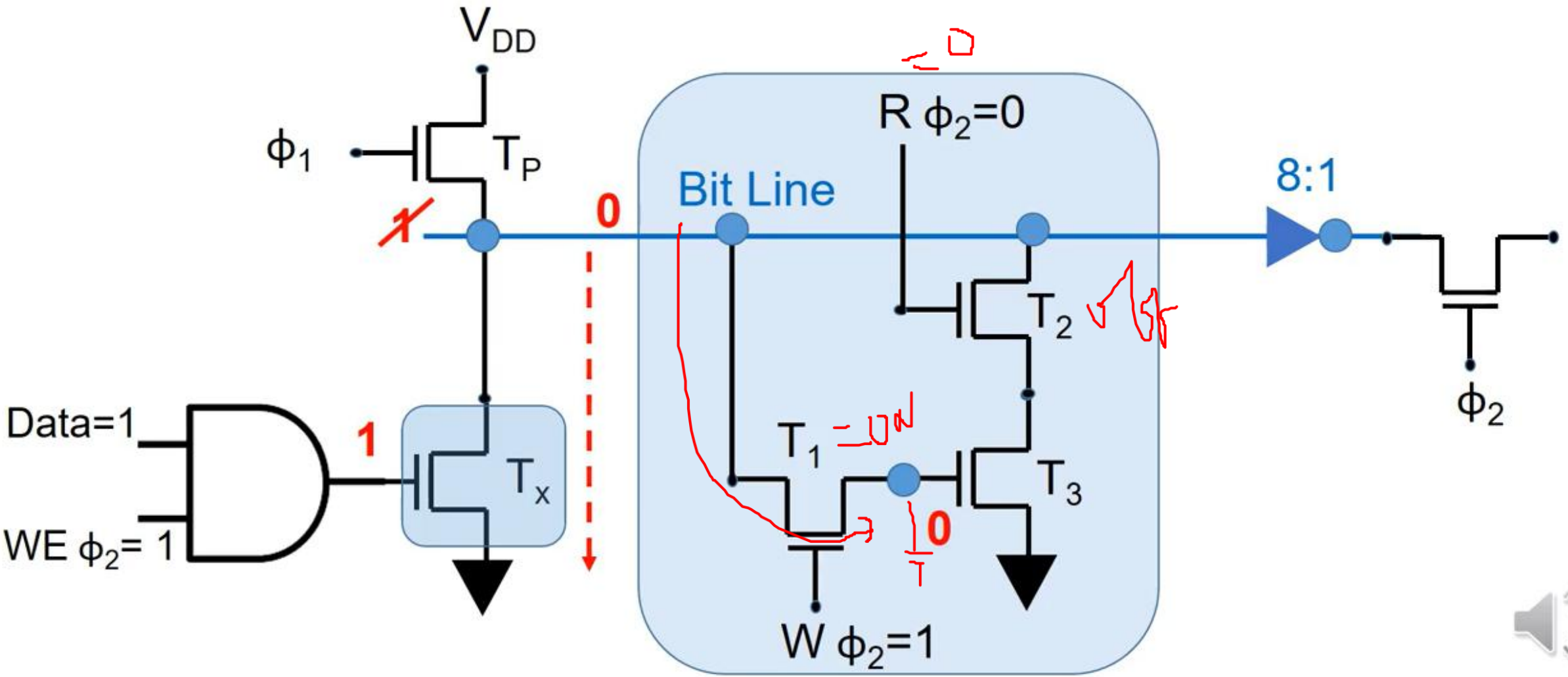
Write 1

Write Operation



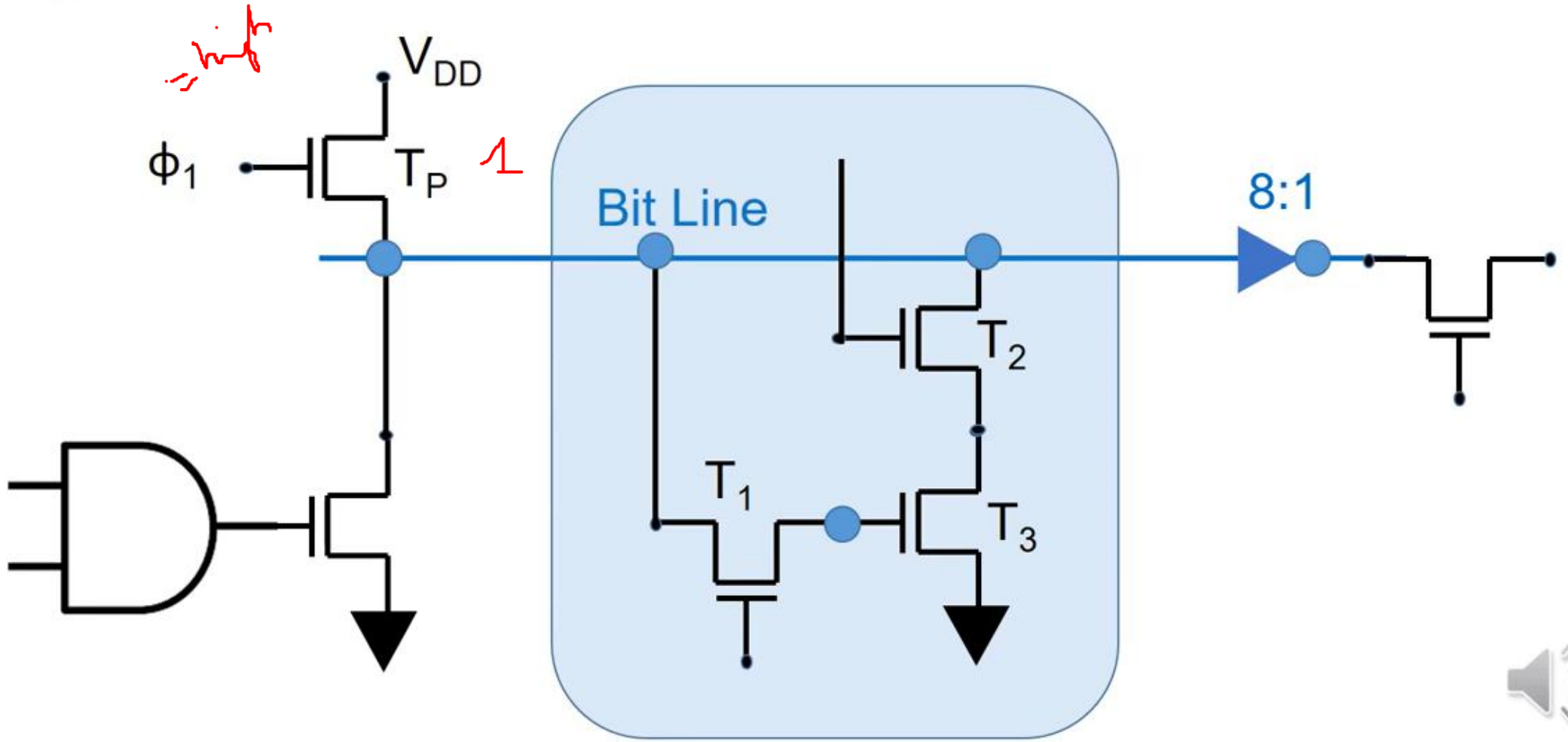
Write 1

Write Operation



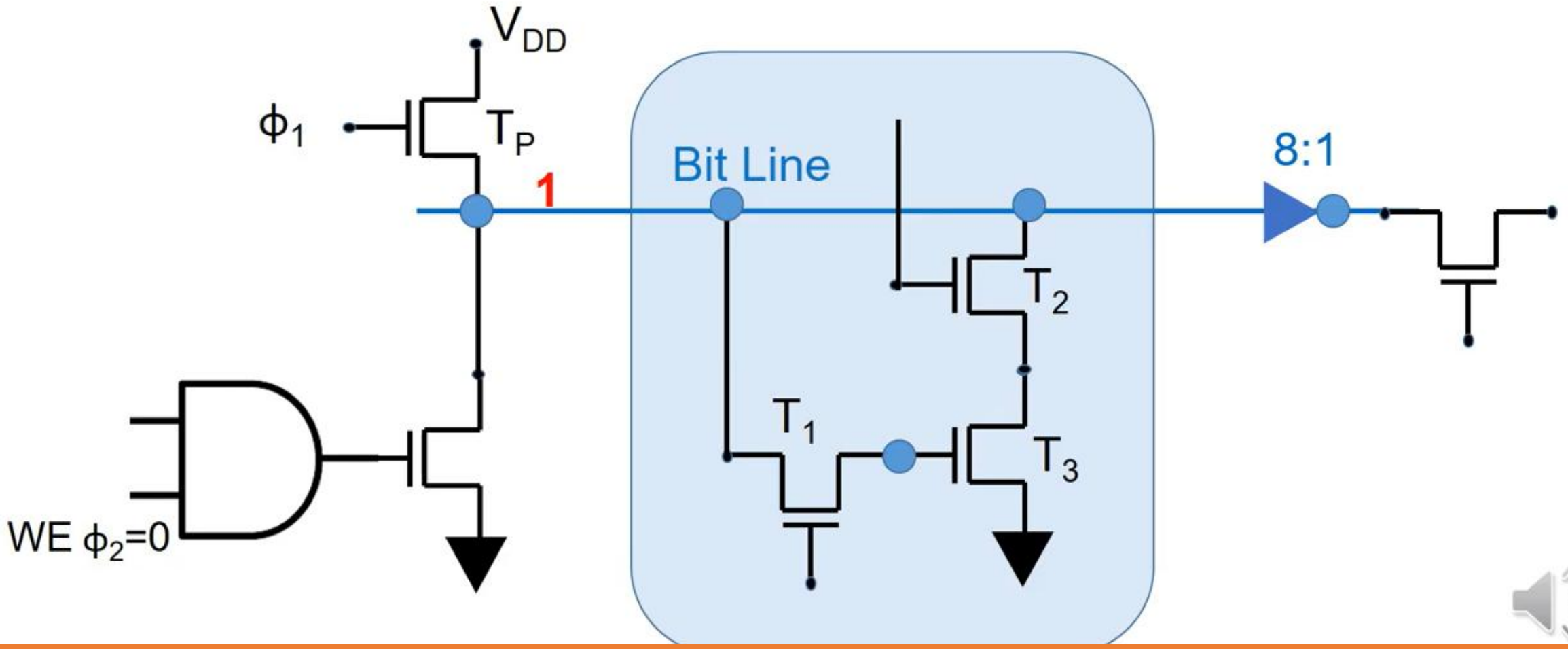
Read 0

Read Operation



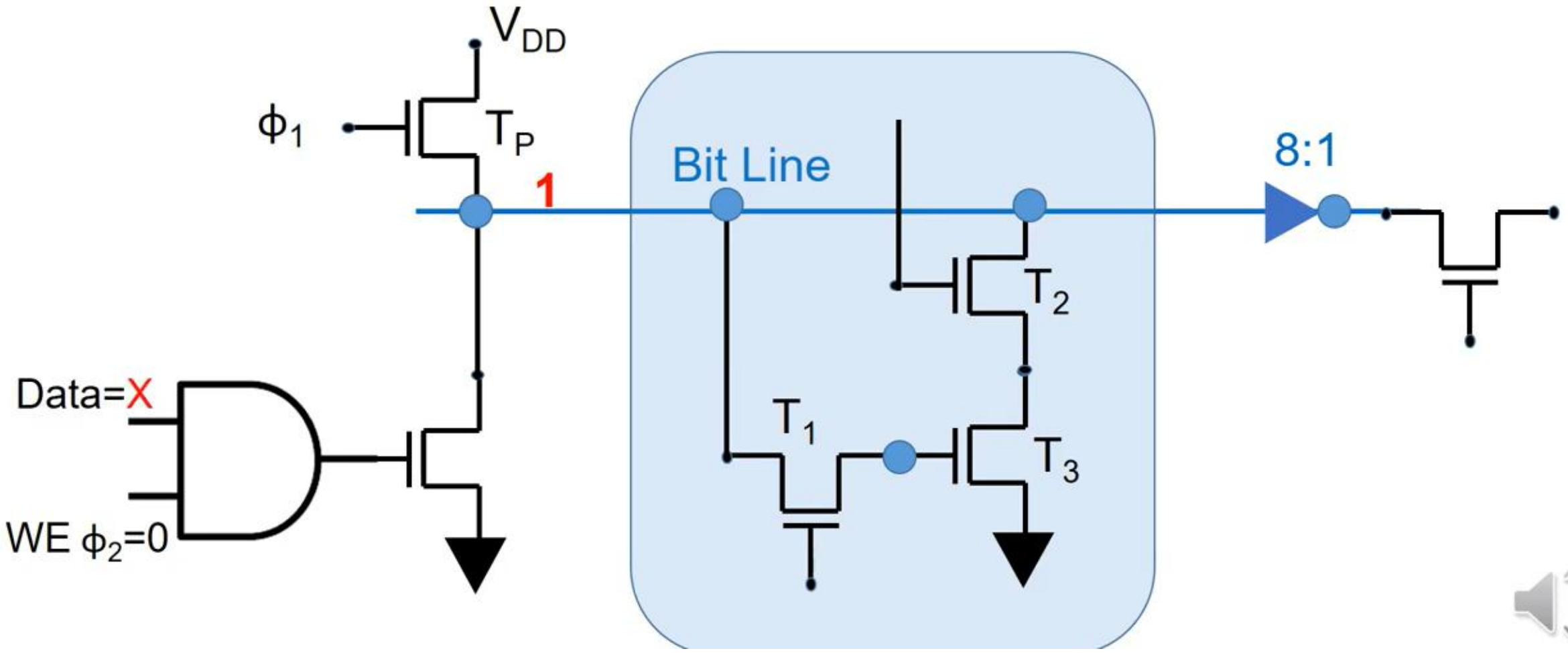
Read 0

Read Operation



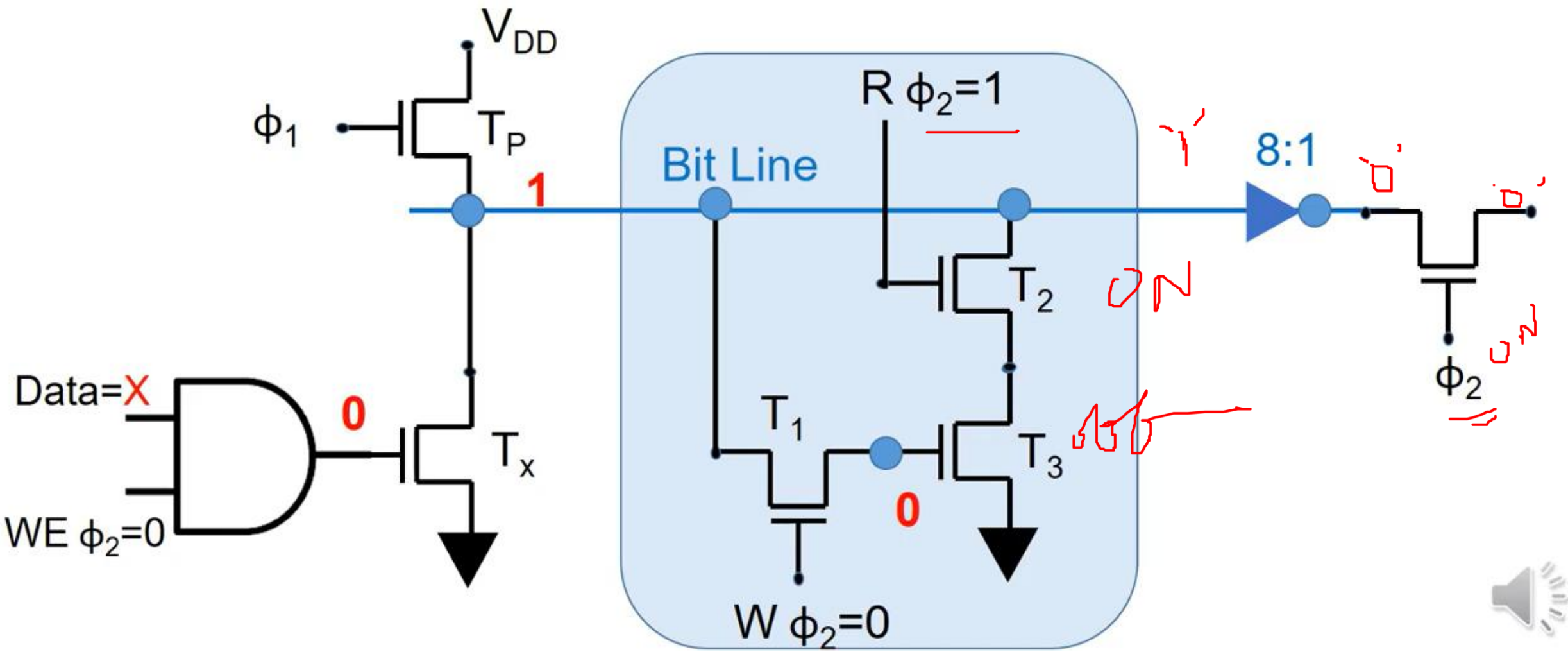
Read 0

Read Operation



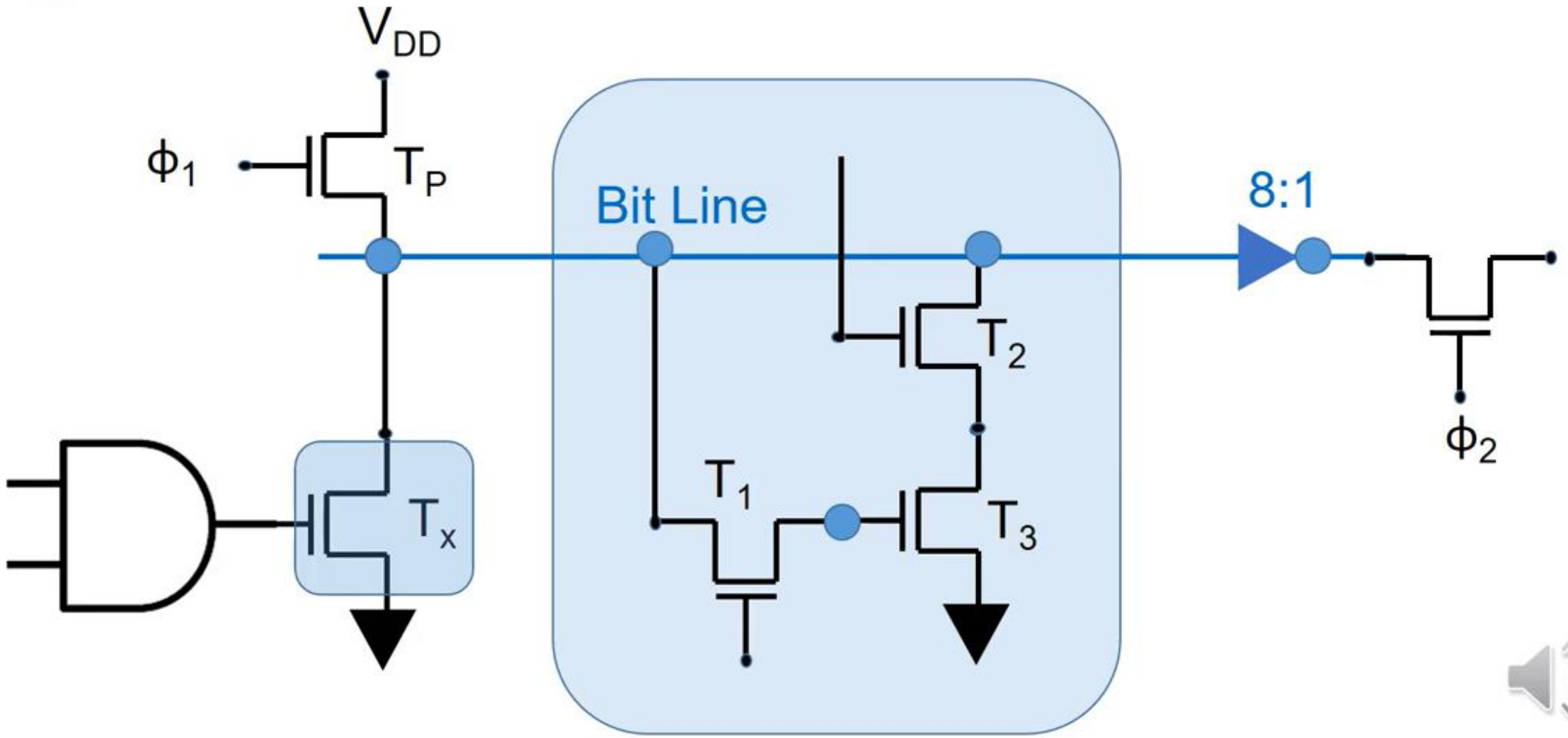
Read Operation

Read 0



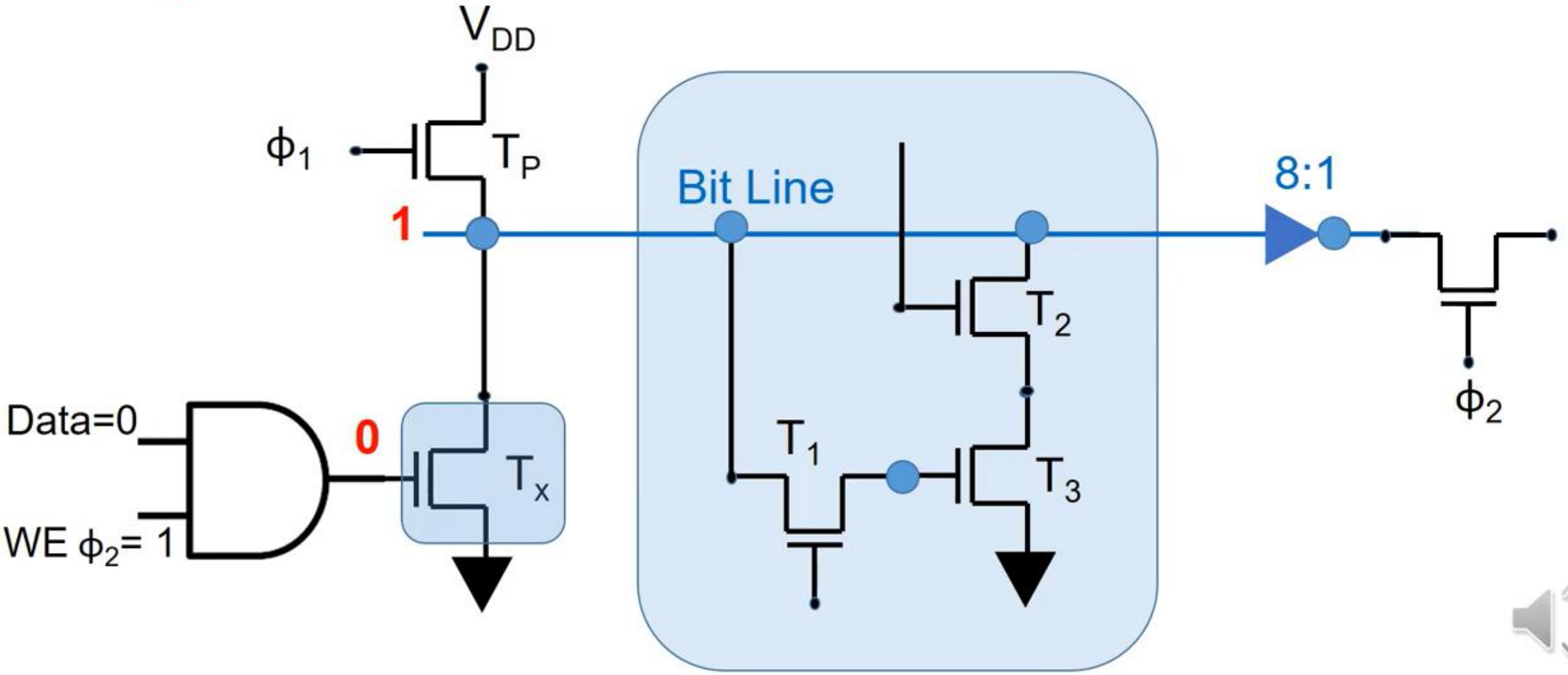
Write 0

Write Operation



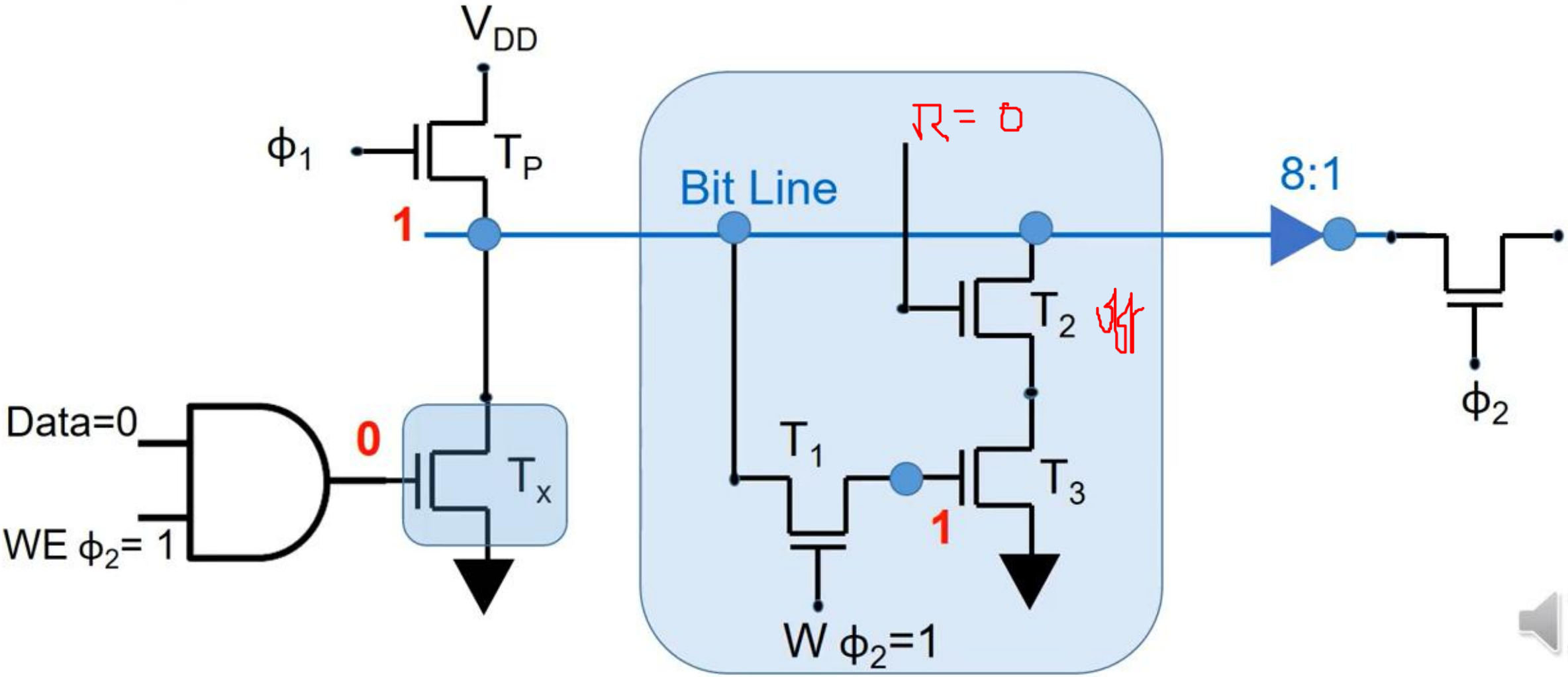
Write 0

Write Operation



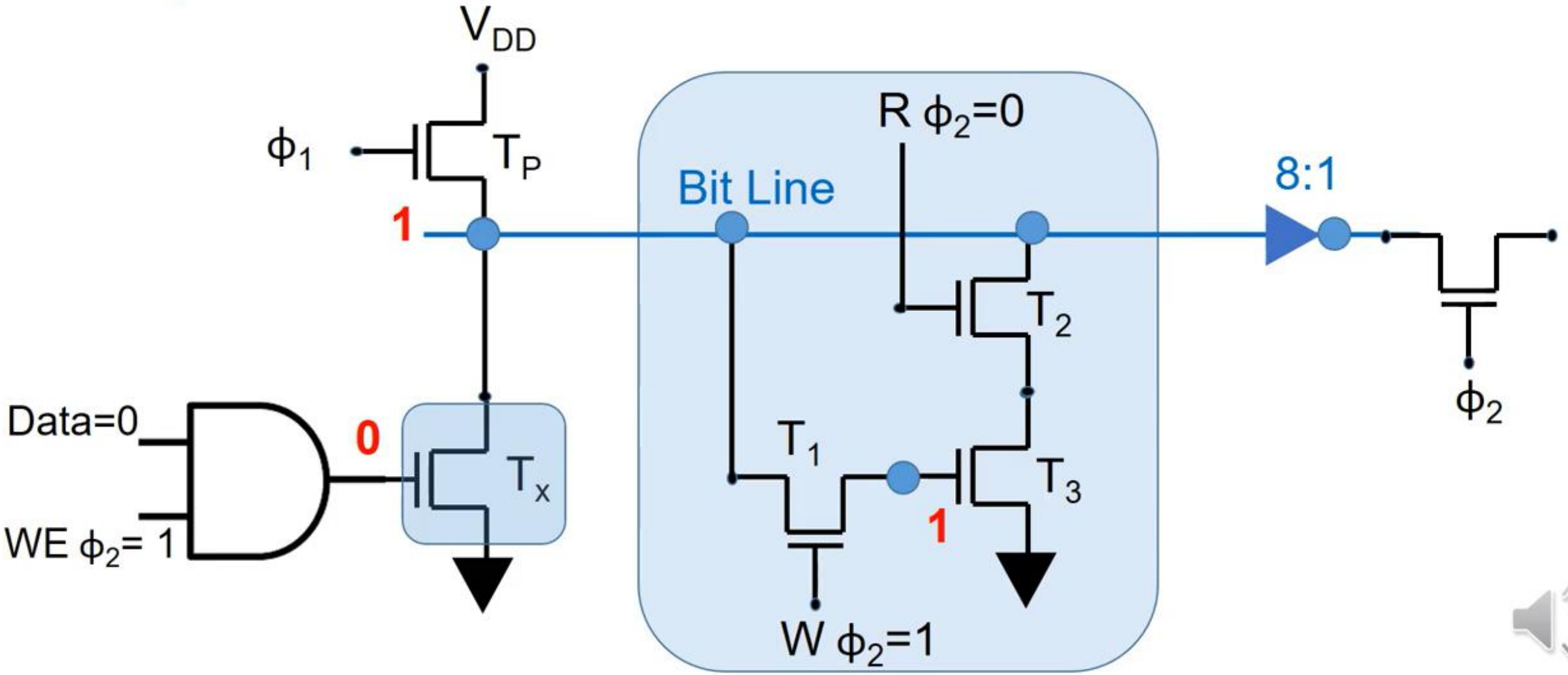
Write 0

Write Operation



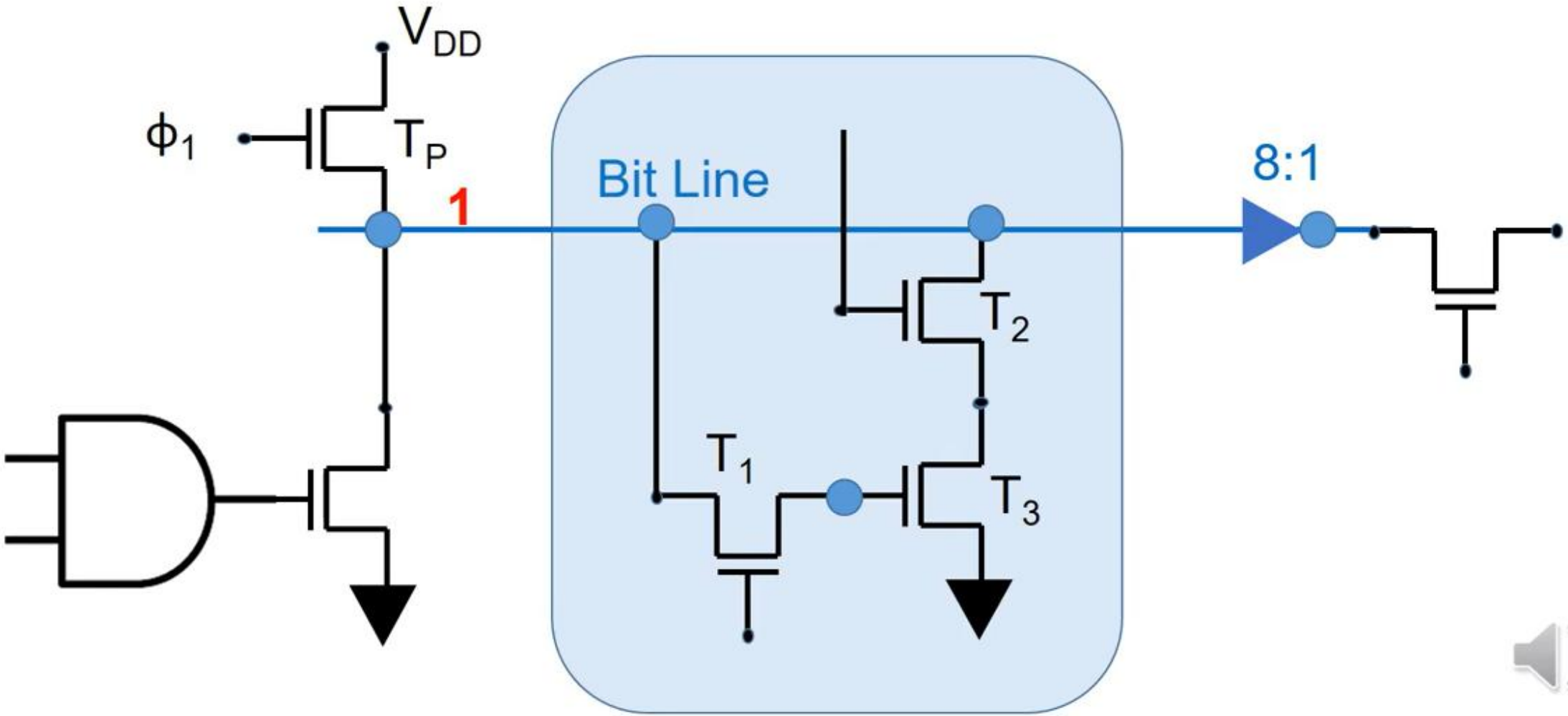
Write 0

Write Operation



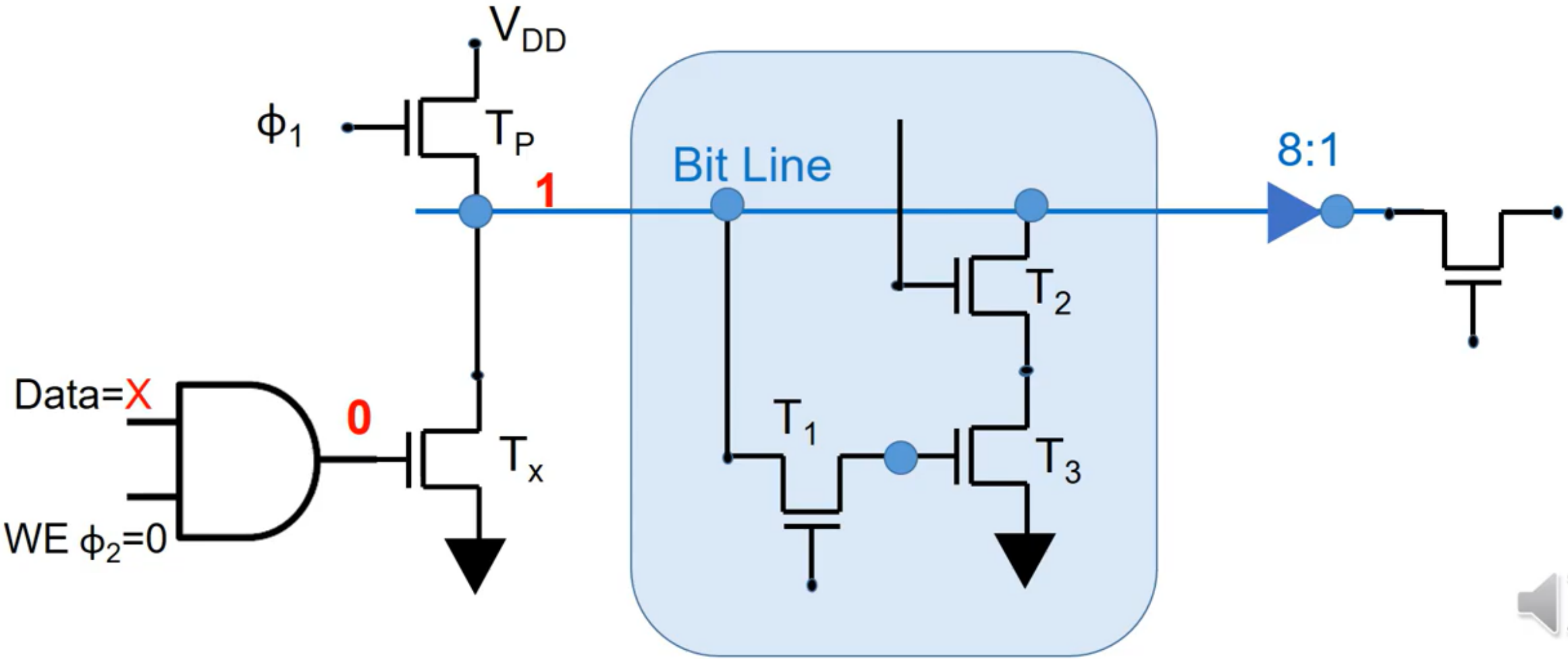
Read Operation

Read 1



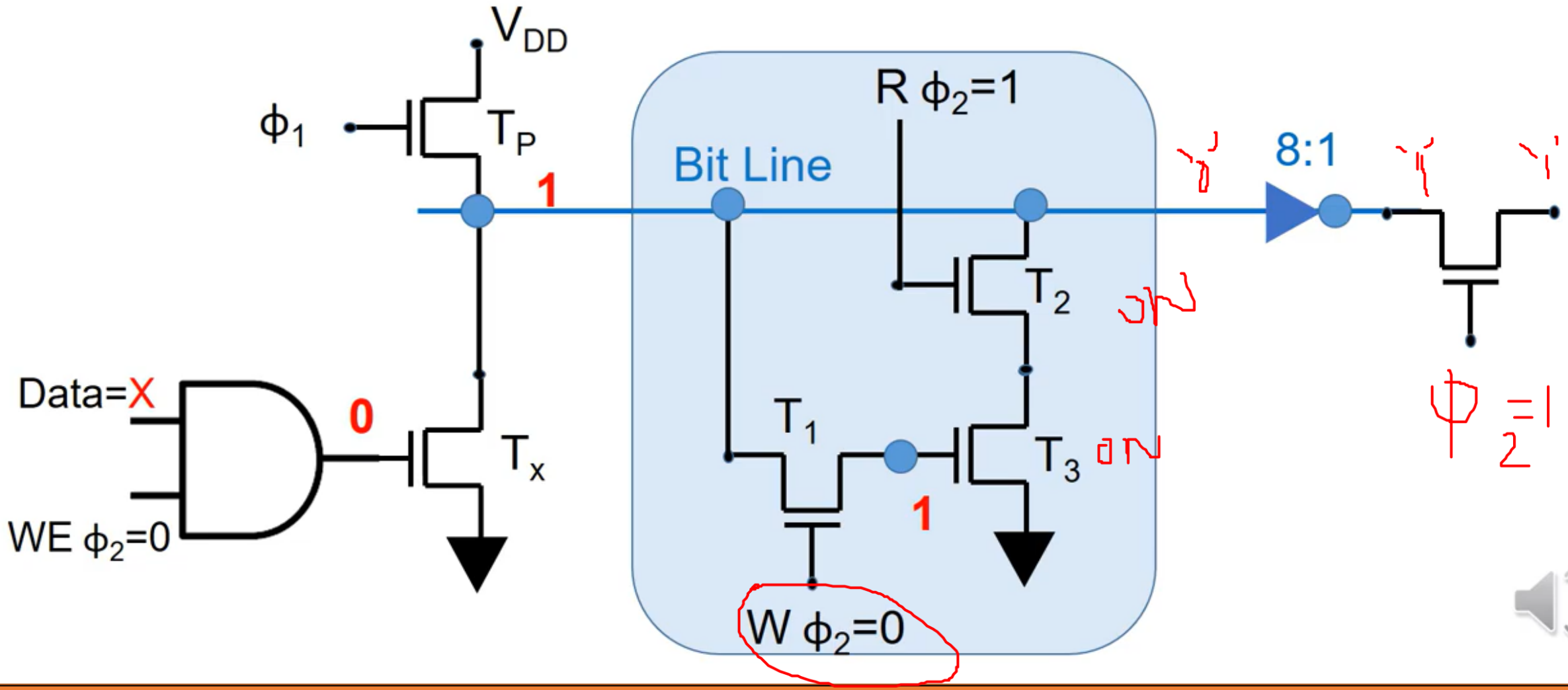
Read 1

Read Operation

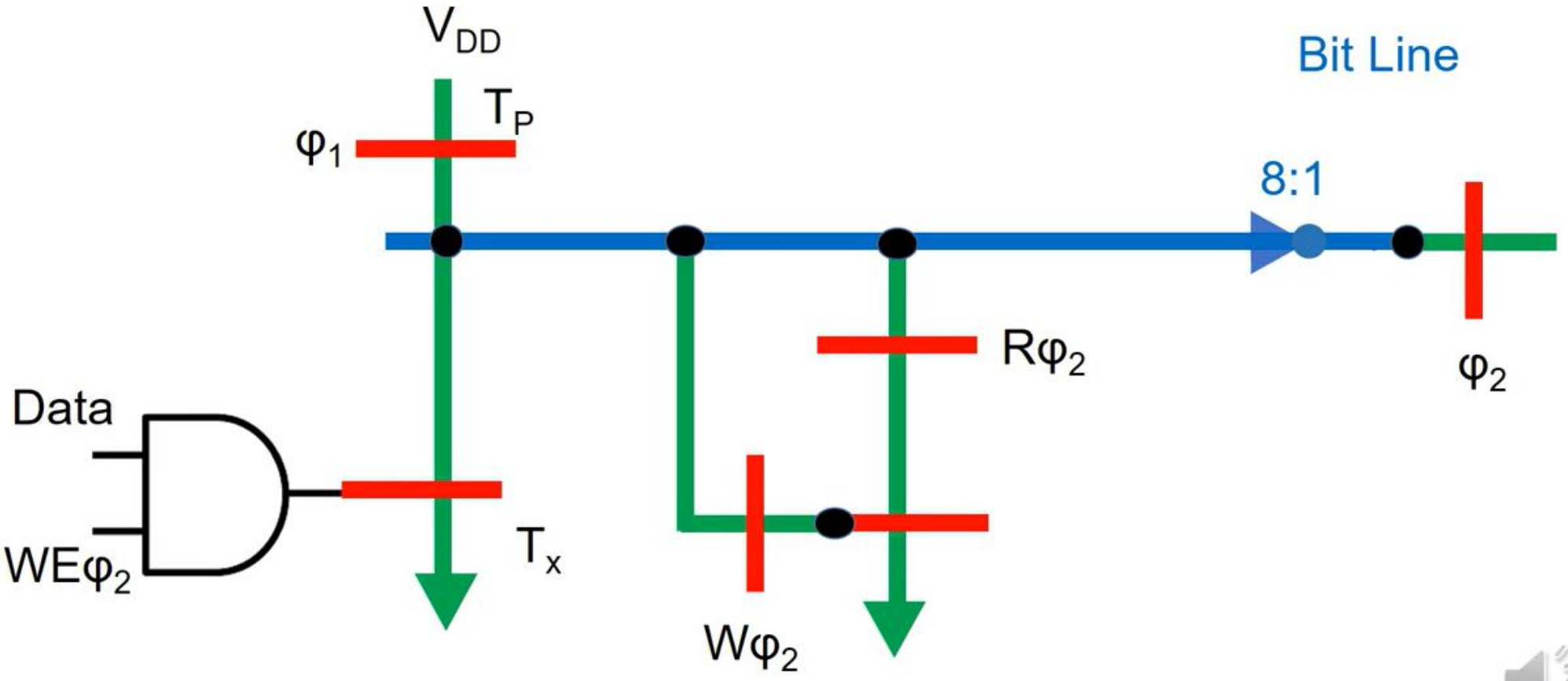


Read 1

Read Operation

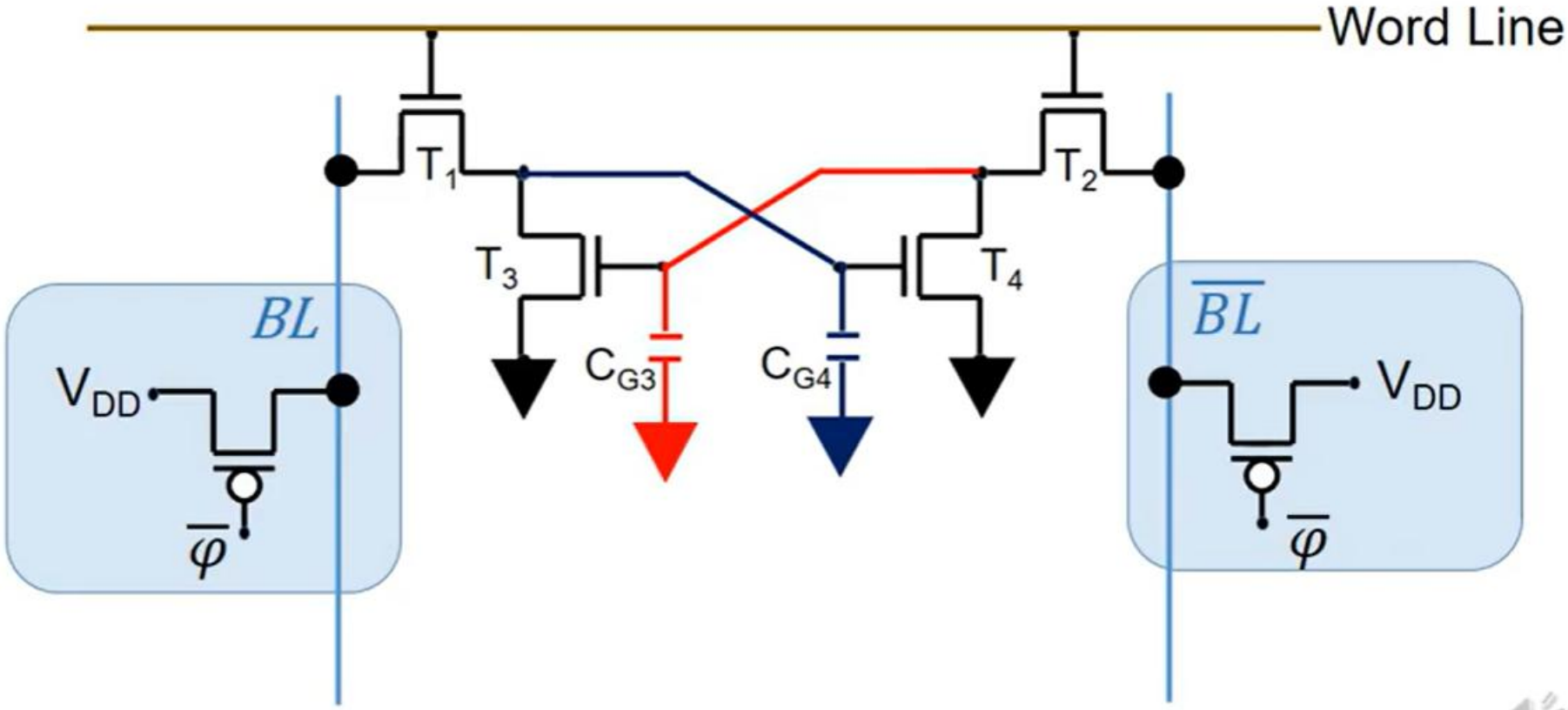


Random Access Memory (3T-DRAM)

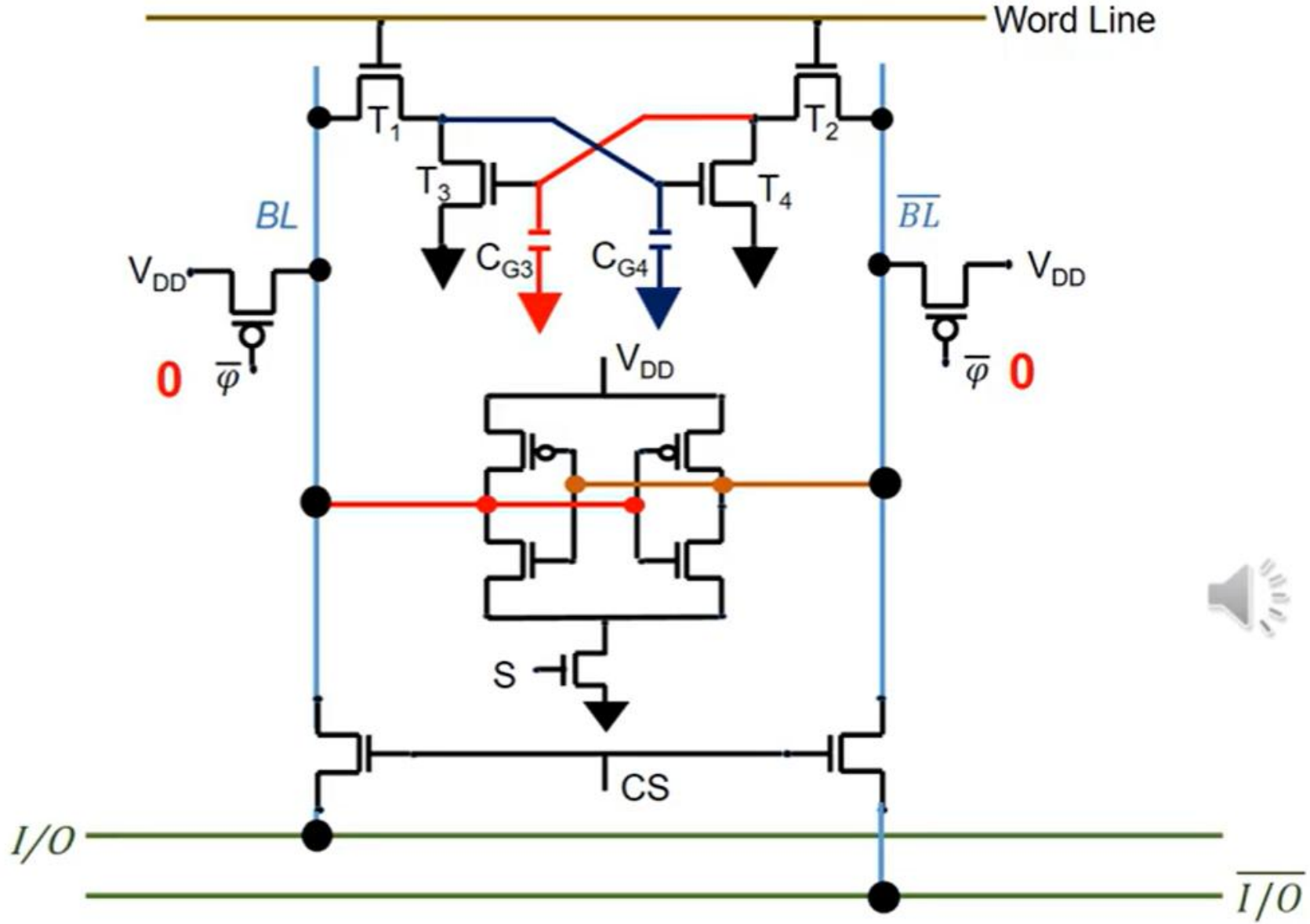


4T-DRAM (4 Transistor – Dynamic Random Access Memory)

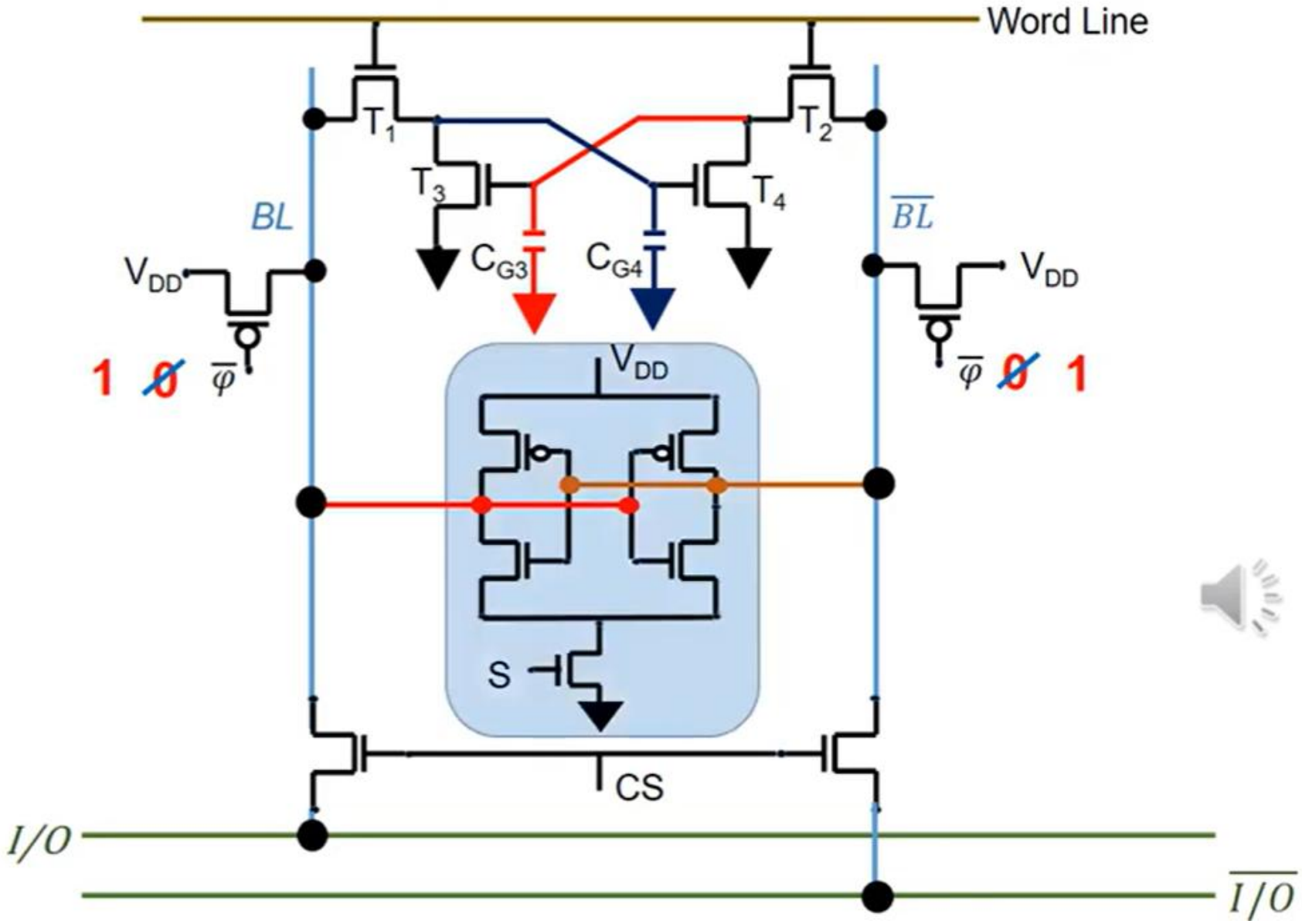
Random Access Memory (4T-DRAM)



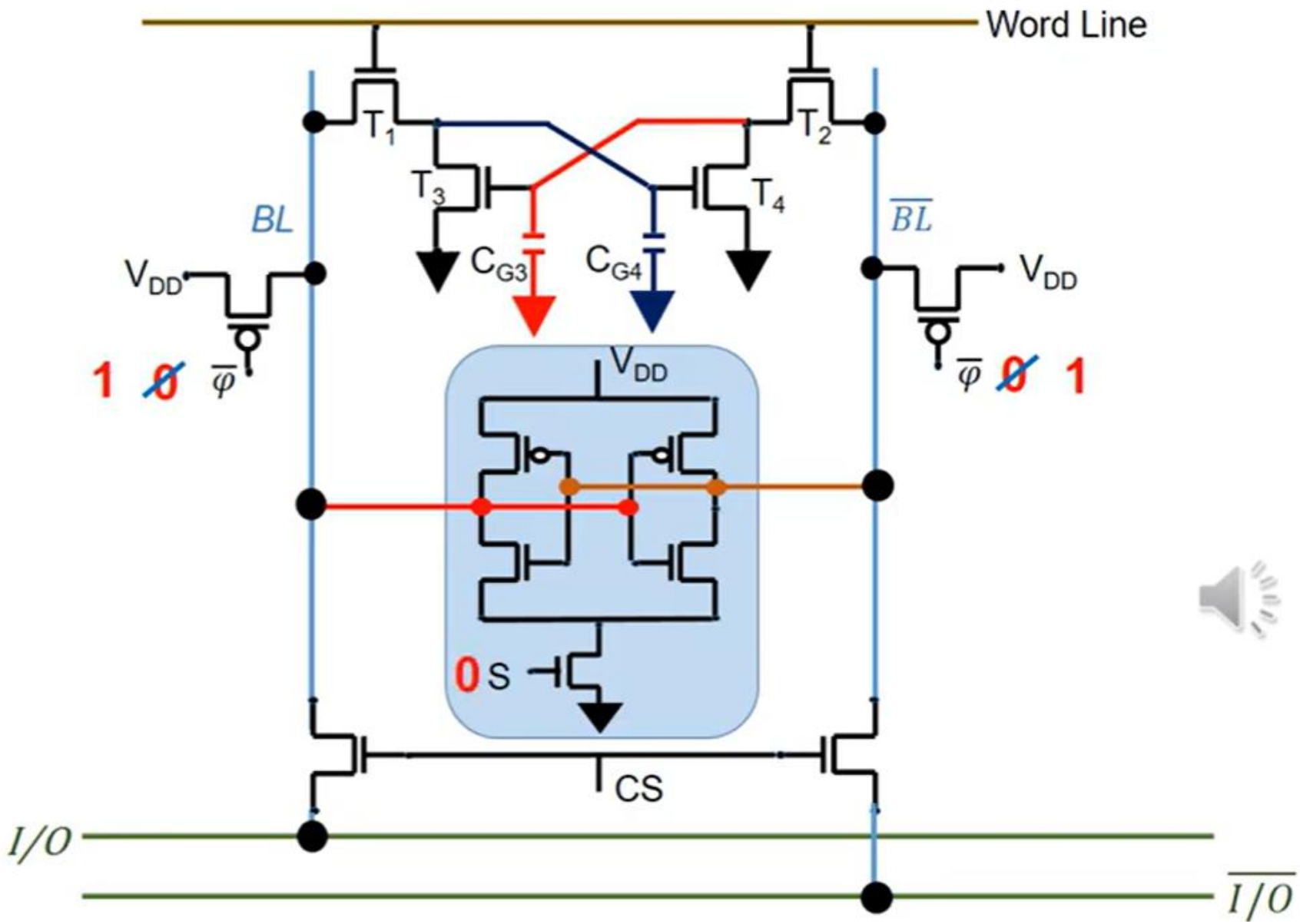
4T-DRAM
Write Operation



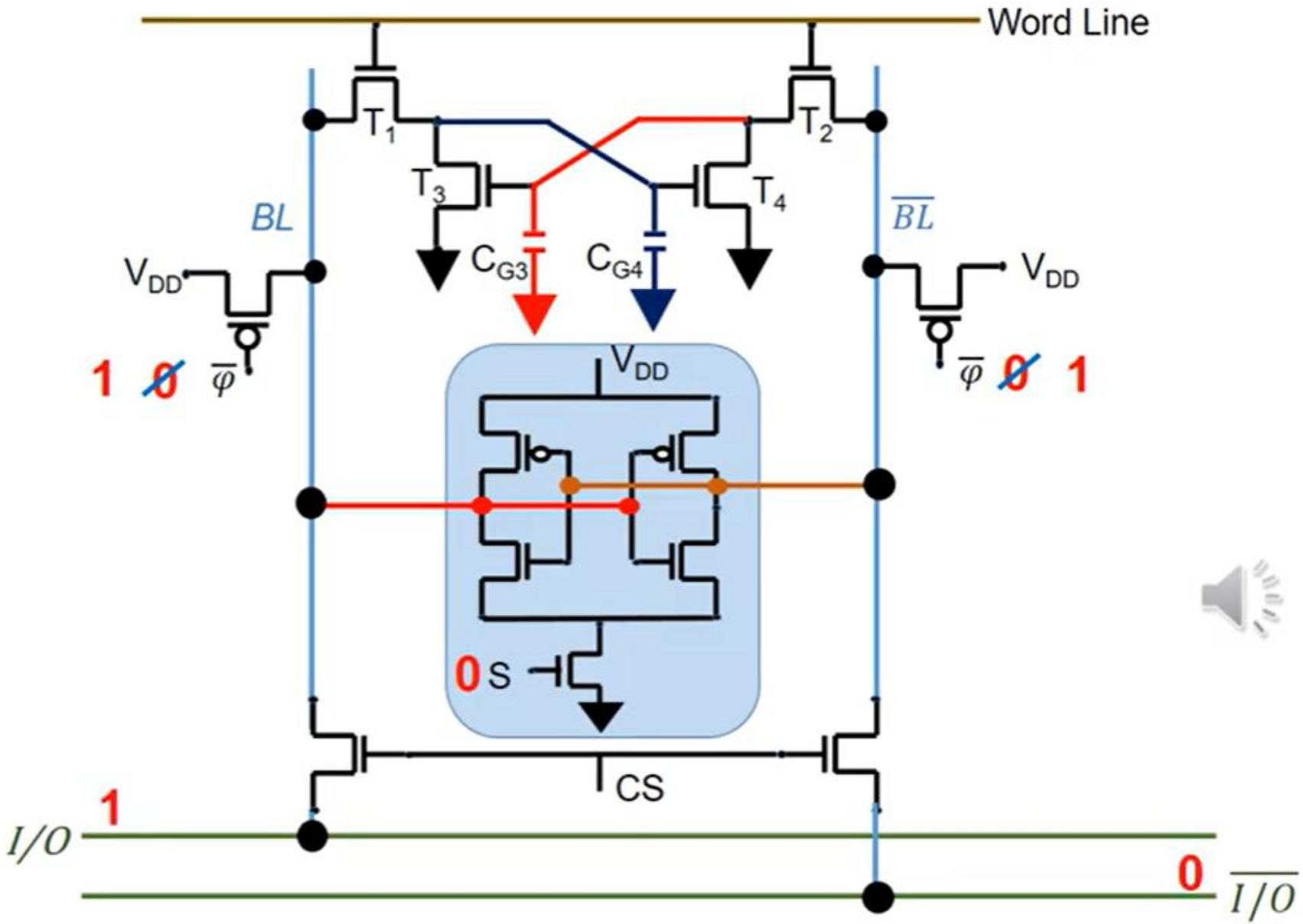
4T-DRAM
Write Operation



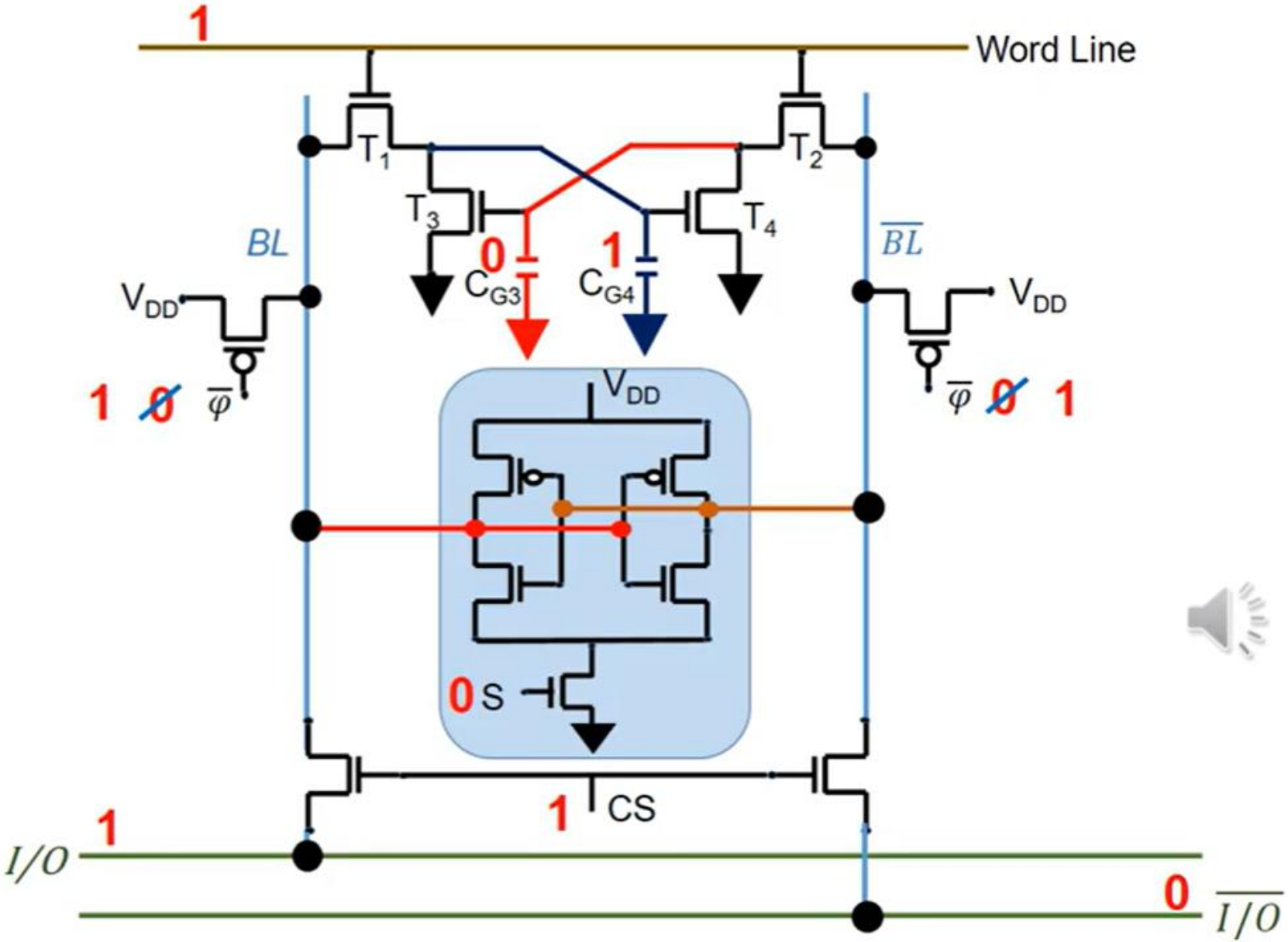
4T-DRAM
Write Operation



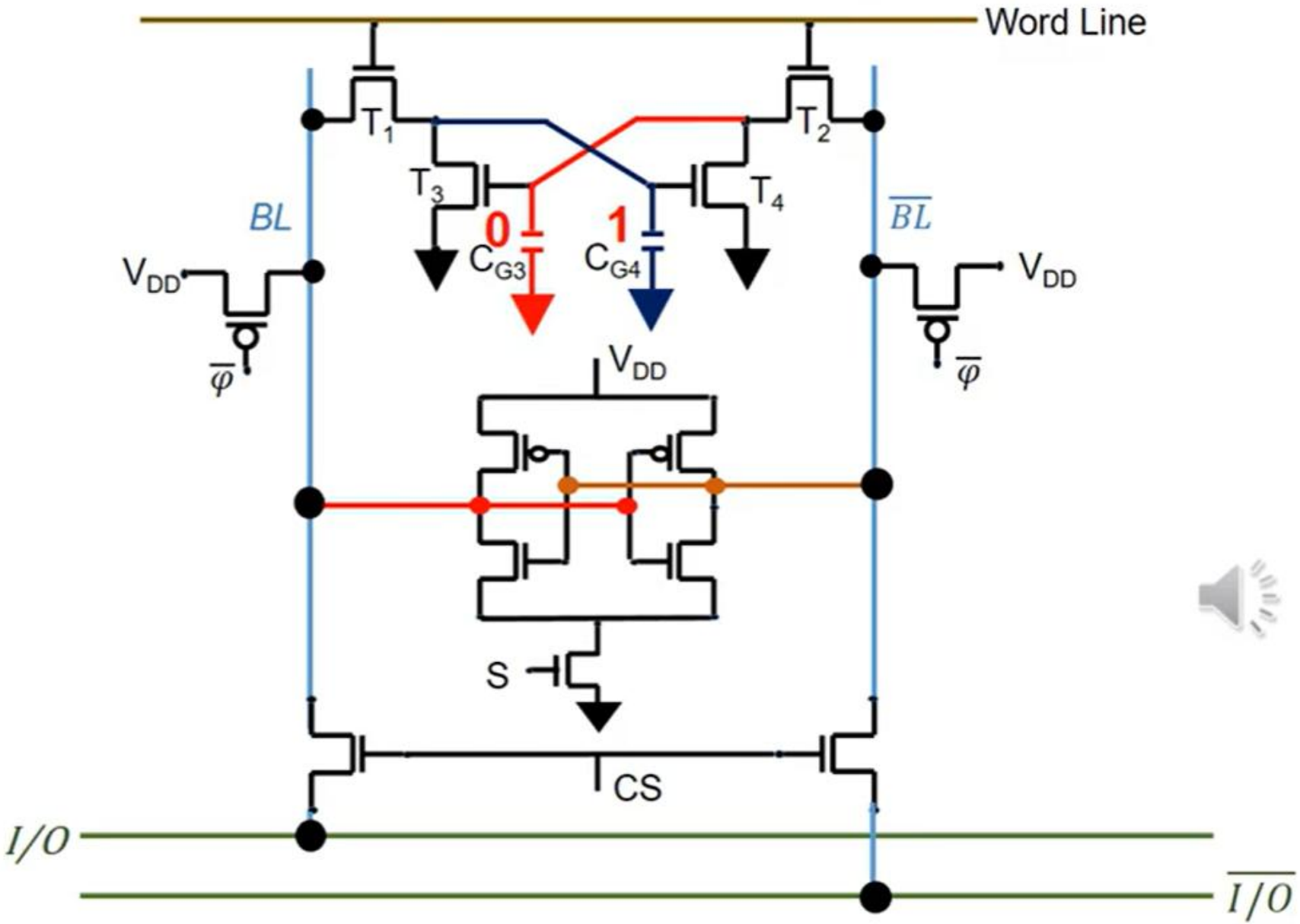
4T-DRAM
Write Operation



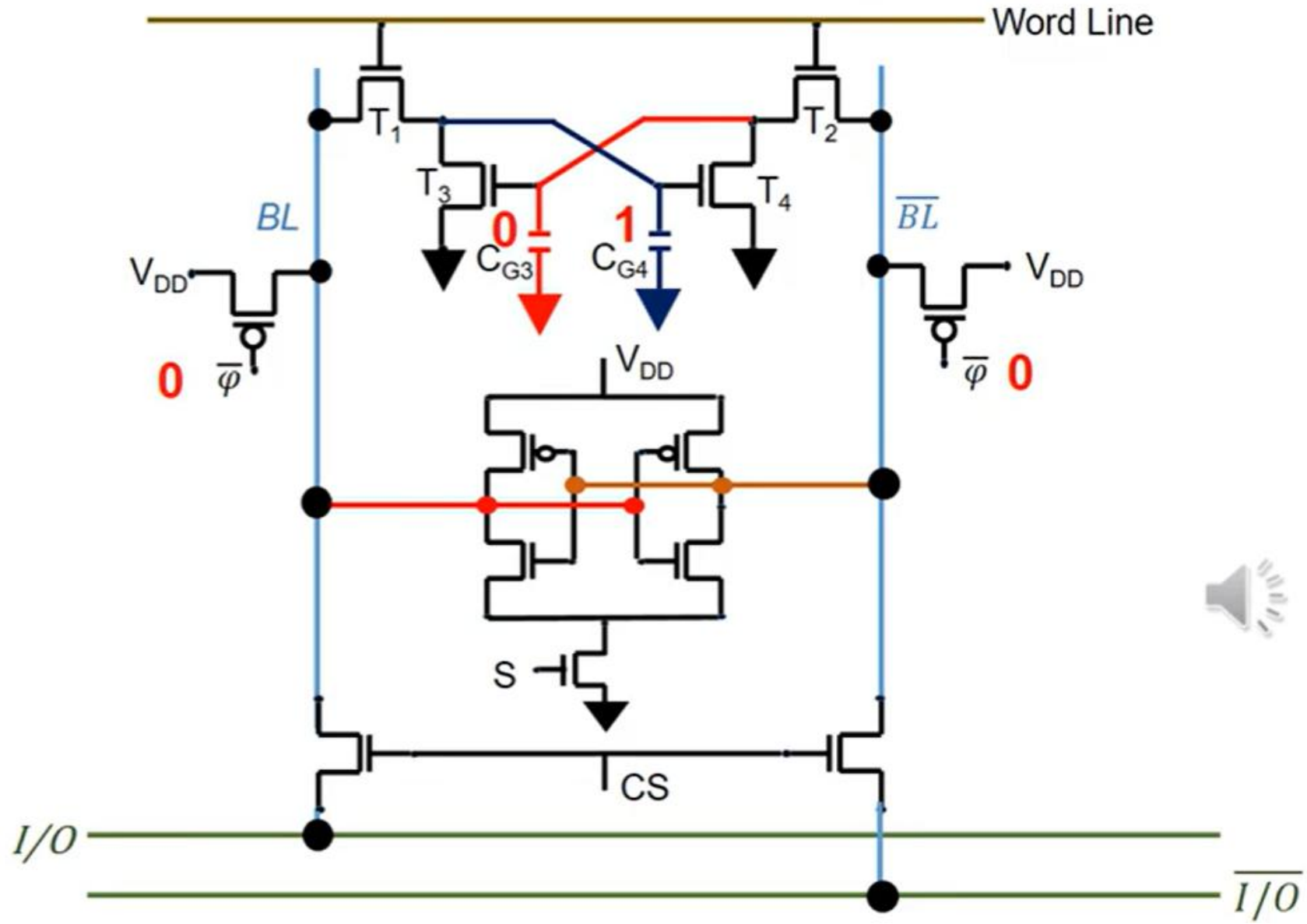
4T-DRAM
Write Operation



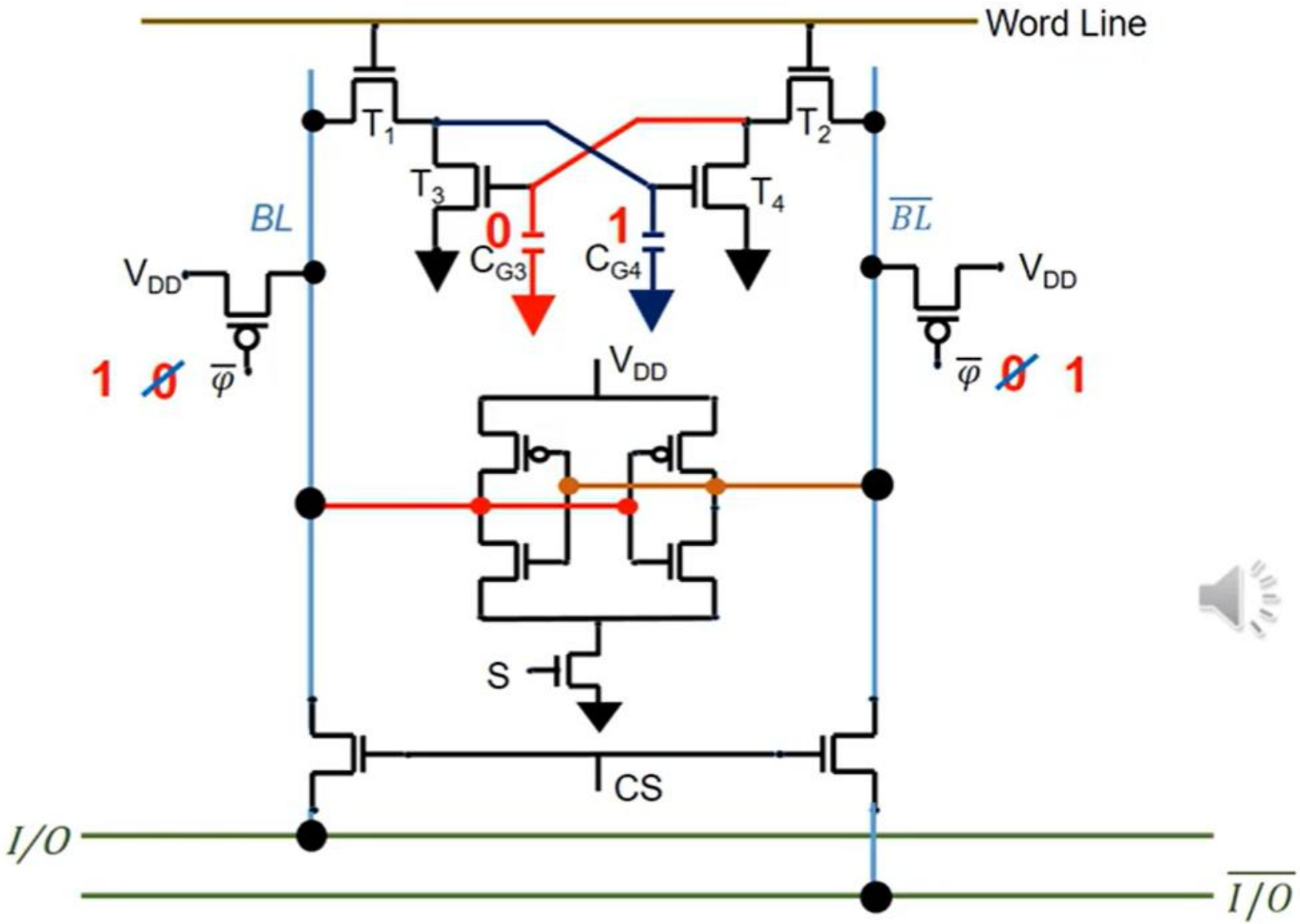
4T-DRAM
Read Operation



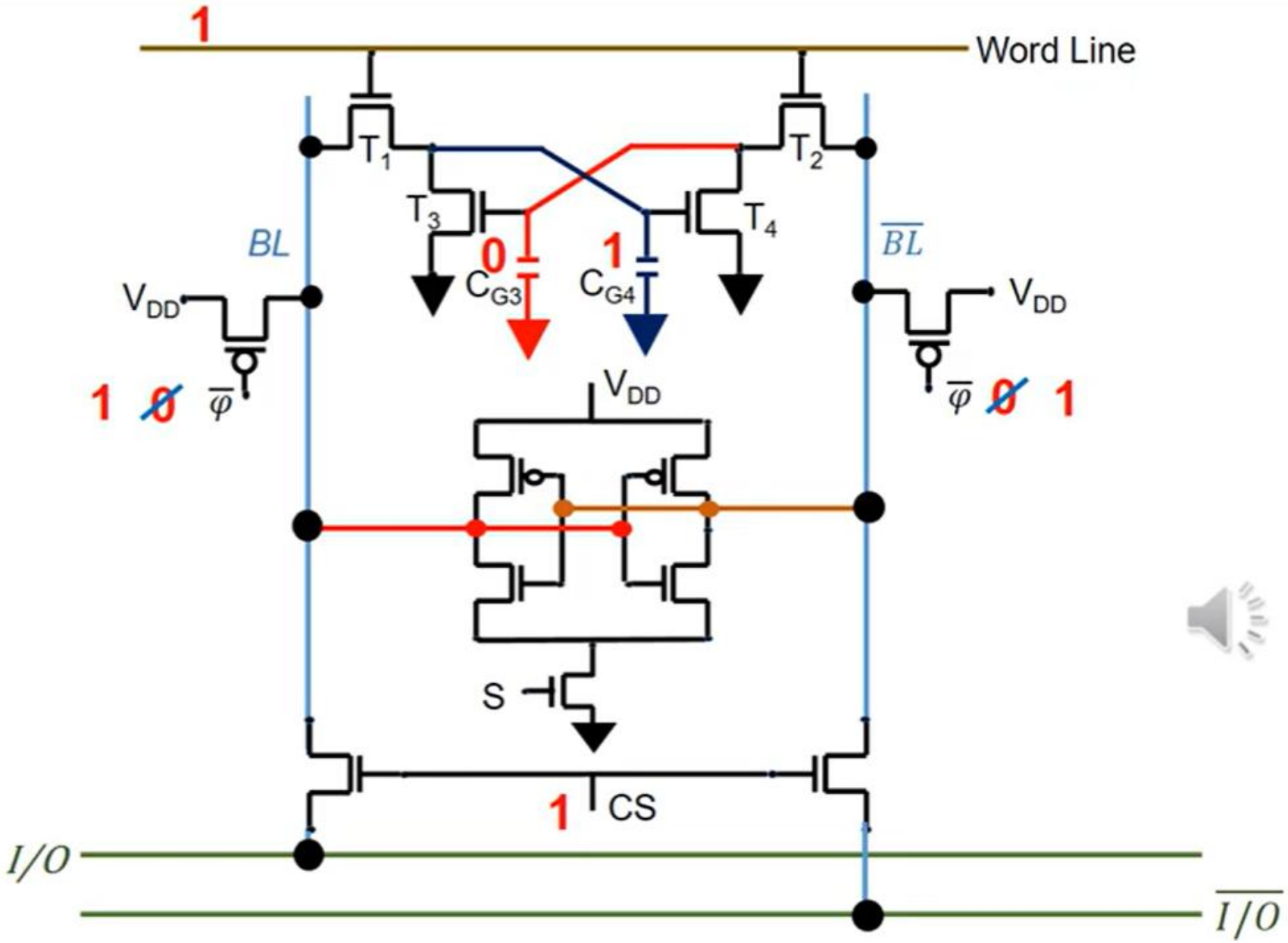
4T-DRAM
Read Operation



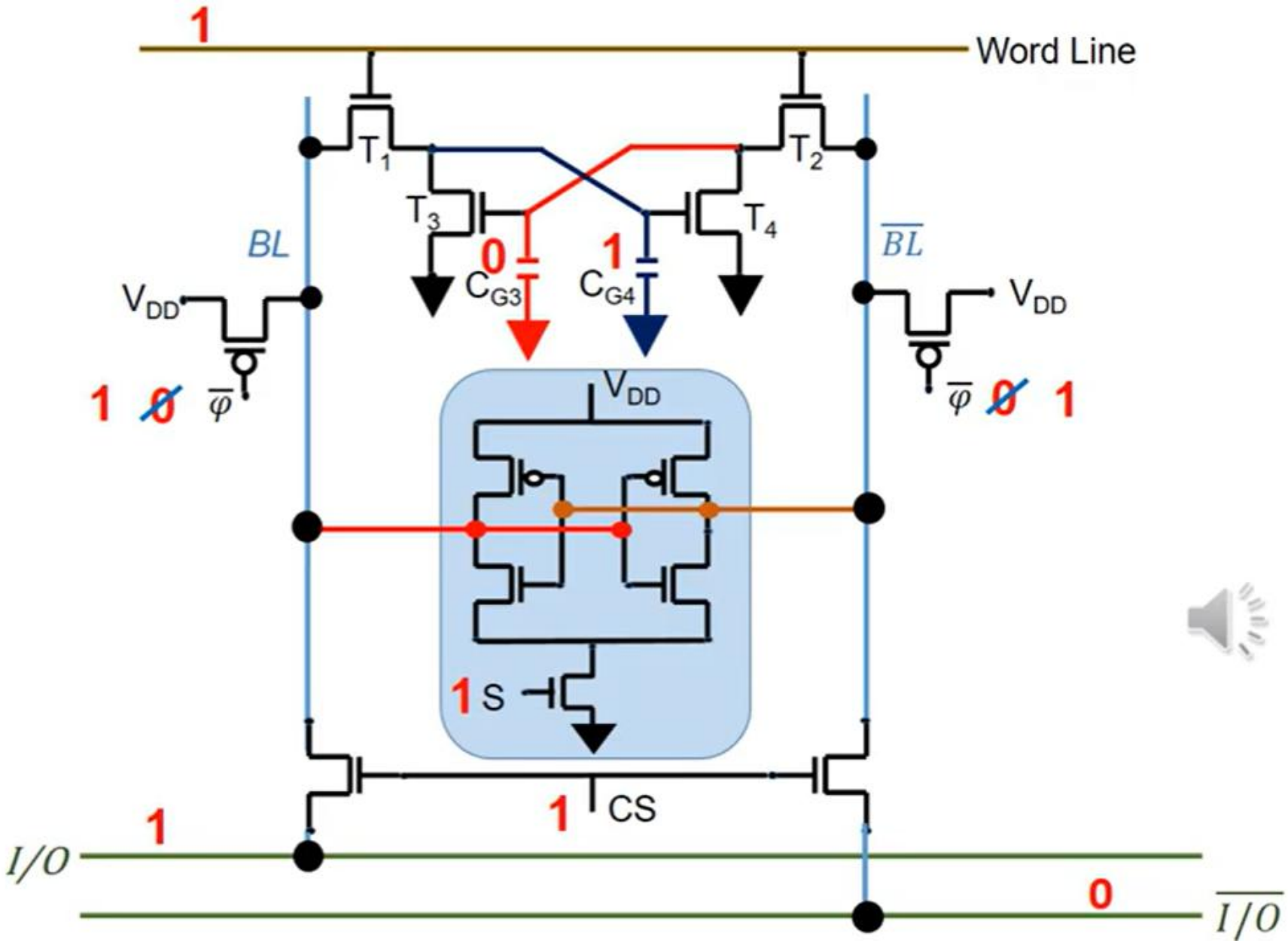
4T-DRAM
Read Operation



4T-DRAM
Read Operation



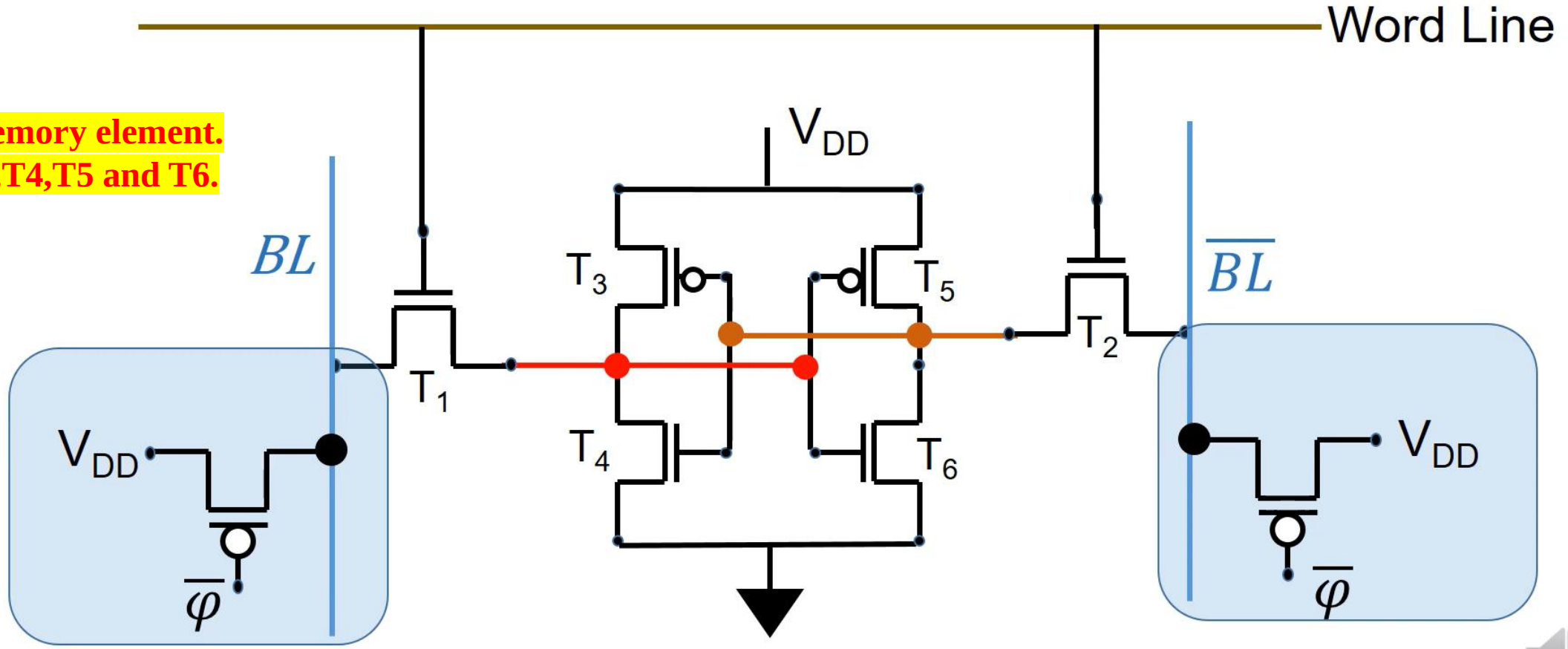
4T-DRAM
Read Operation



6T-SRAM (6 Transistor – Static Random Access Memory)

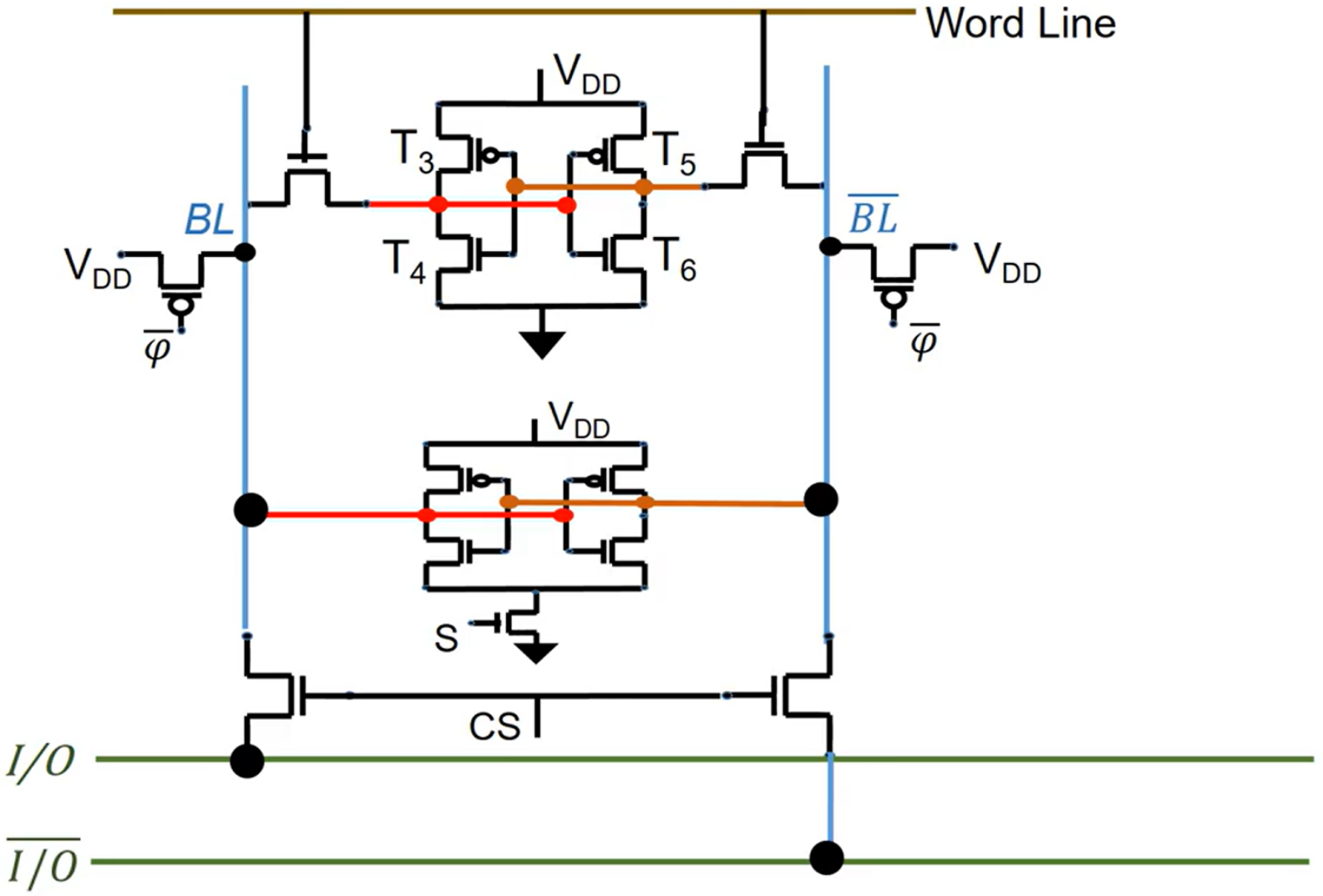
Random Access Memory (6T-SRAM)

Bi-stable memory element. made of T3,T4,T5 and T6.



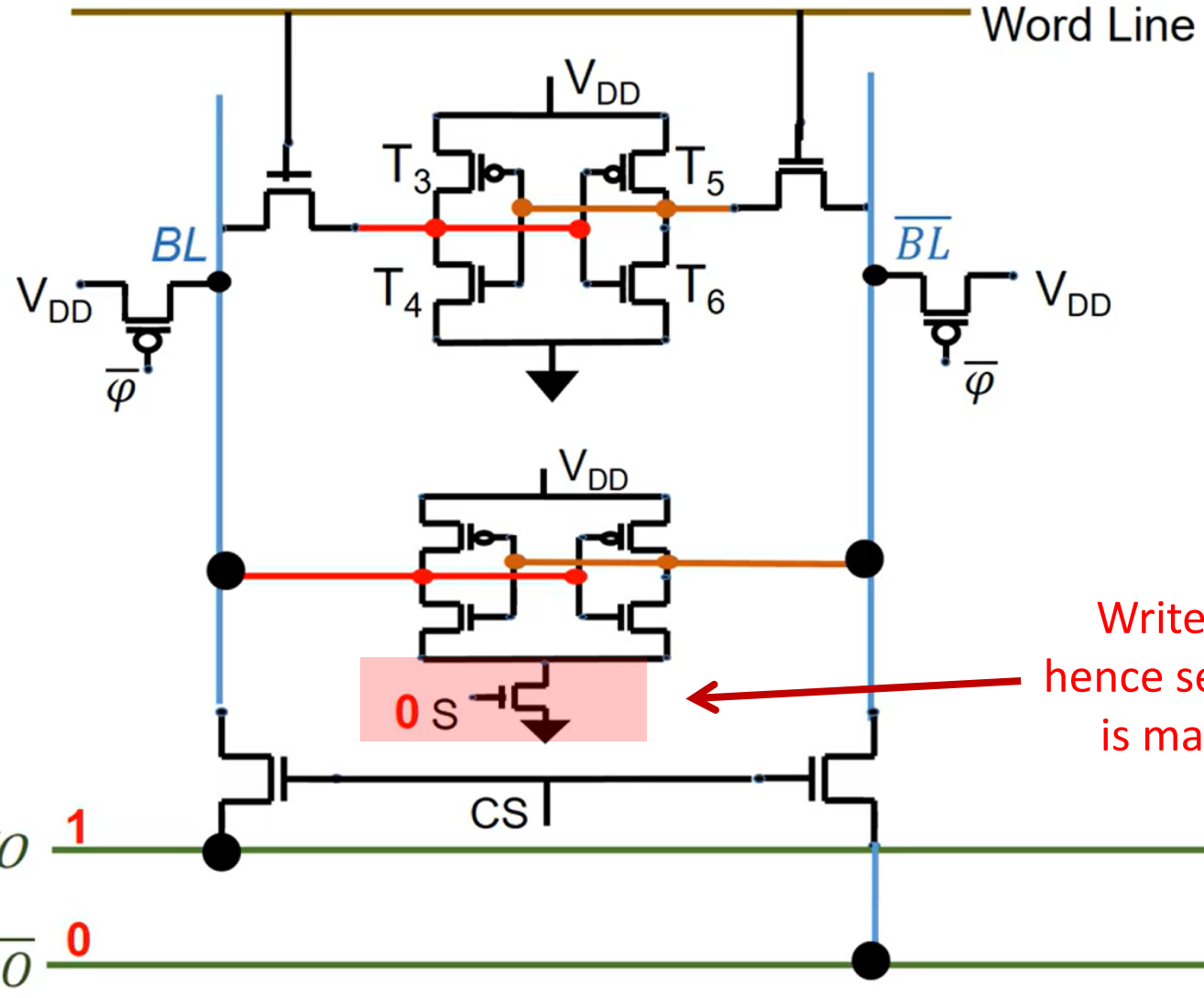
Uses Pre-charge logic, before each Read and Write we need to precharge BL and BL'

Random Access Memory (6T-SRAM)



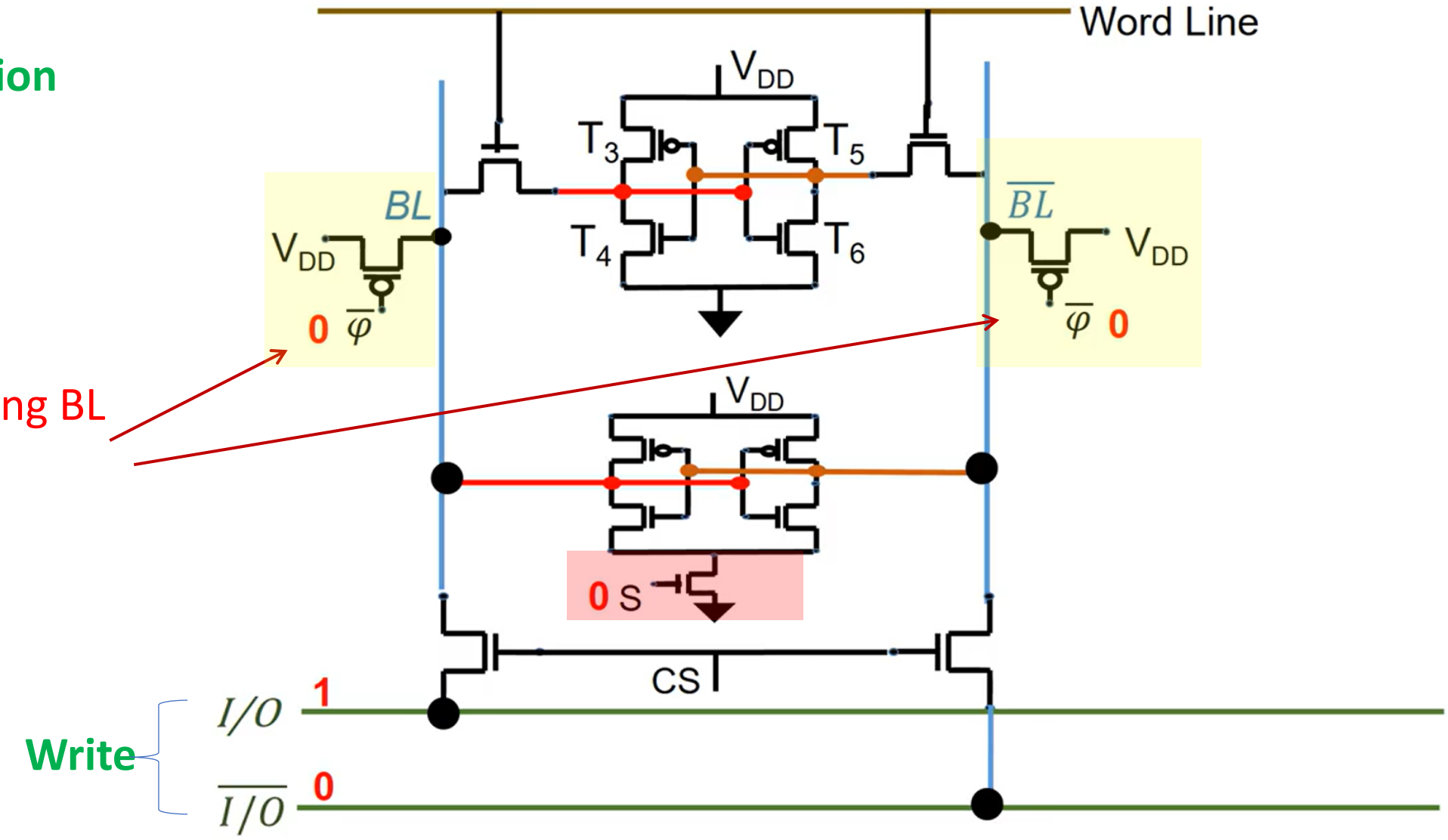
Random Access Memory (6T-SRAM)

Write operation



Random Access Memory (6T-SRAM)

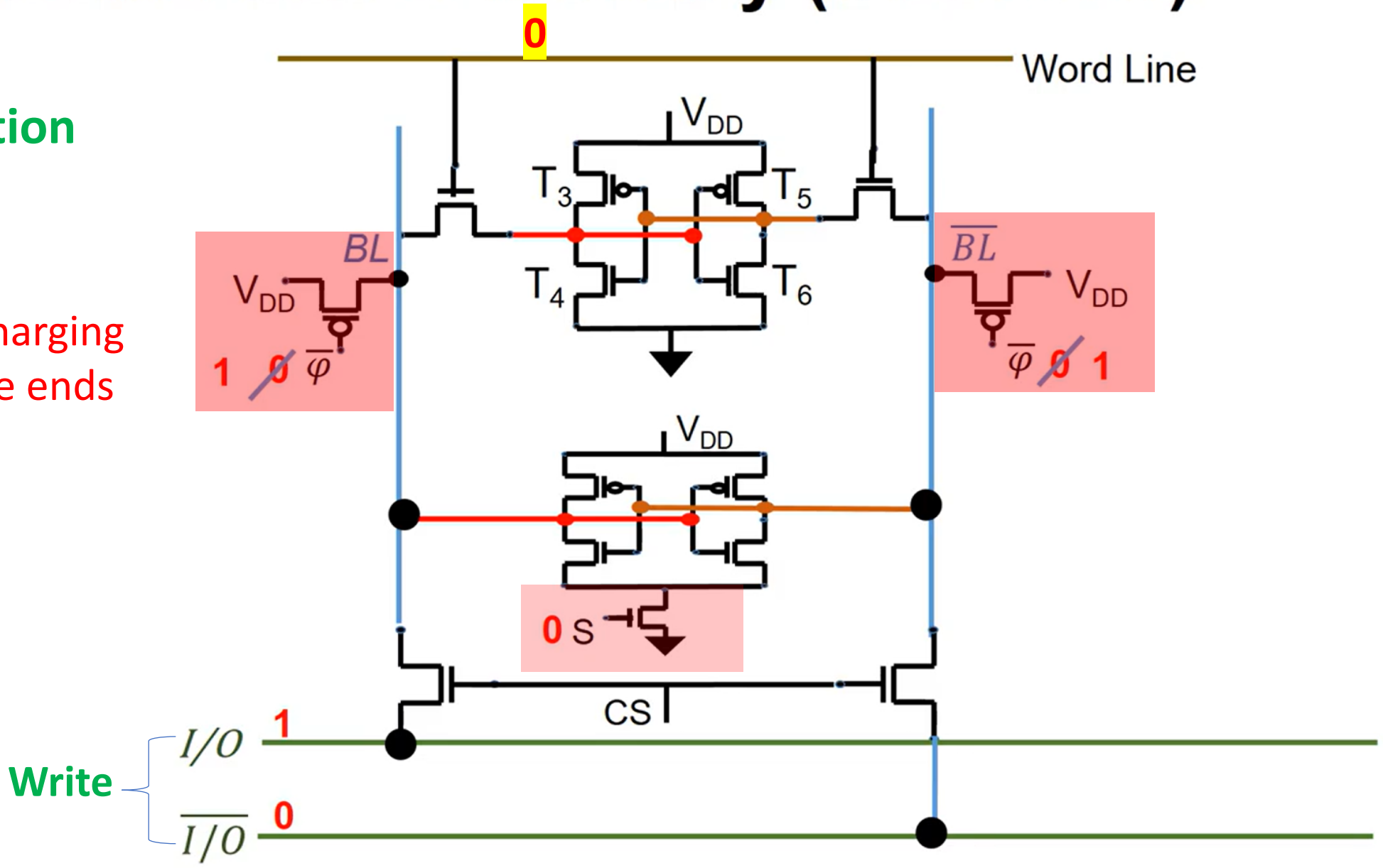
Write operation



Random Access Memory (6T-SRAM)

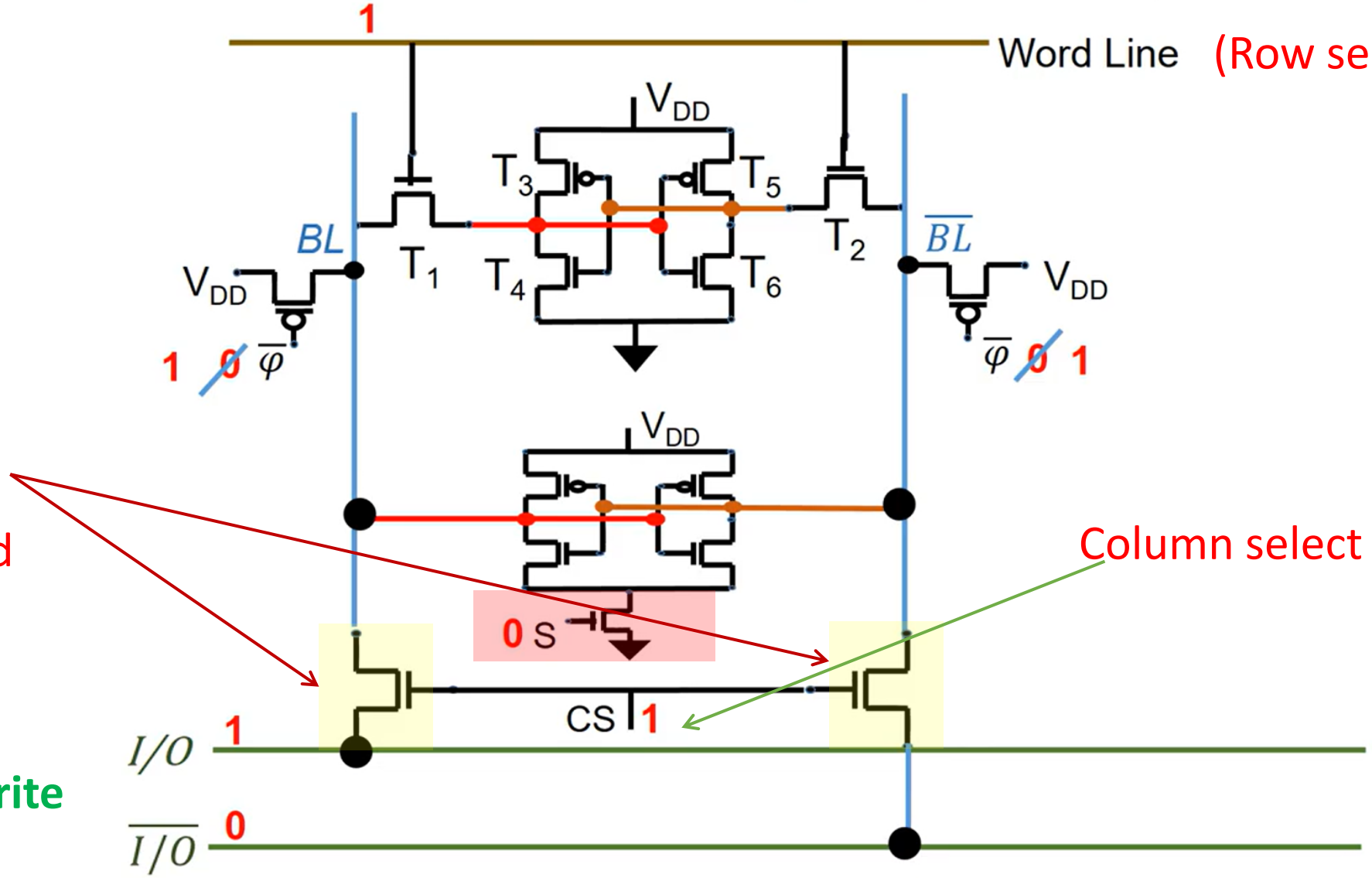
Write operation

Precharging phase ends



Random Access Memory (6T-SRAM)

Write operation



Input to be written is charging BL and discharging BL'

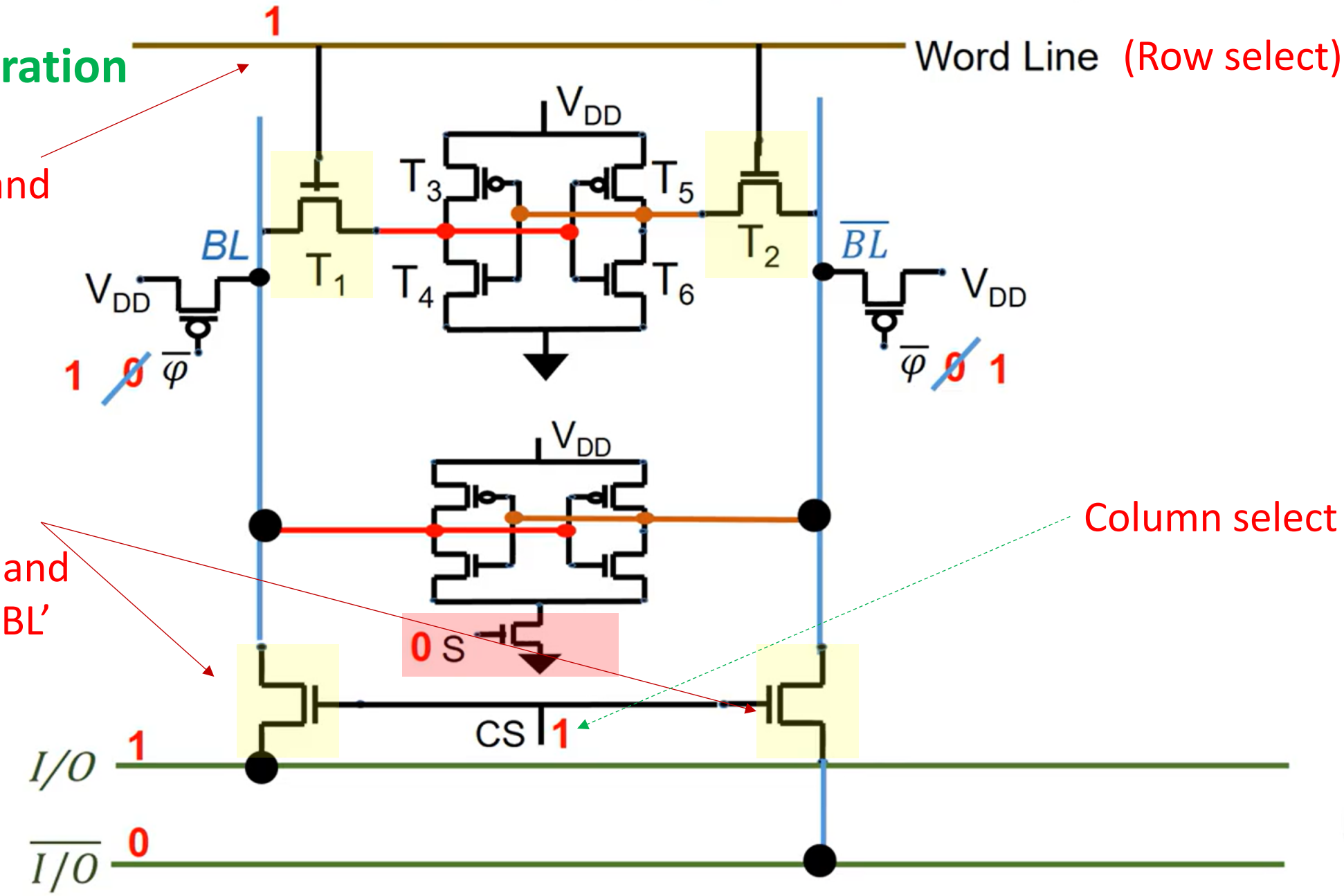
Column select

Write

Random Access Memory (6T-SRAM)

Write operation

Word line is activated, T1 and T2 is ON.

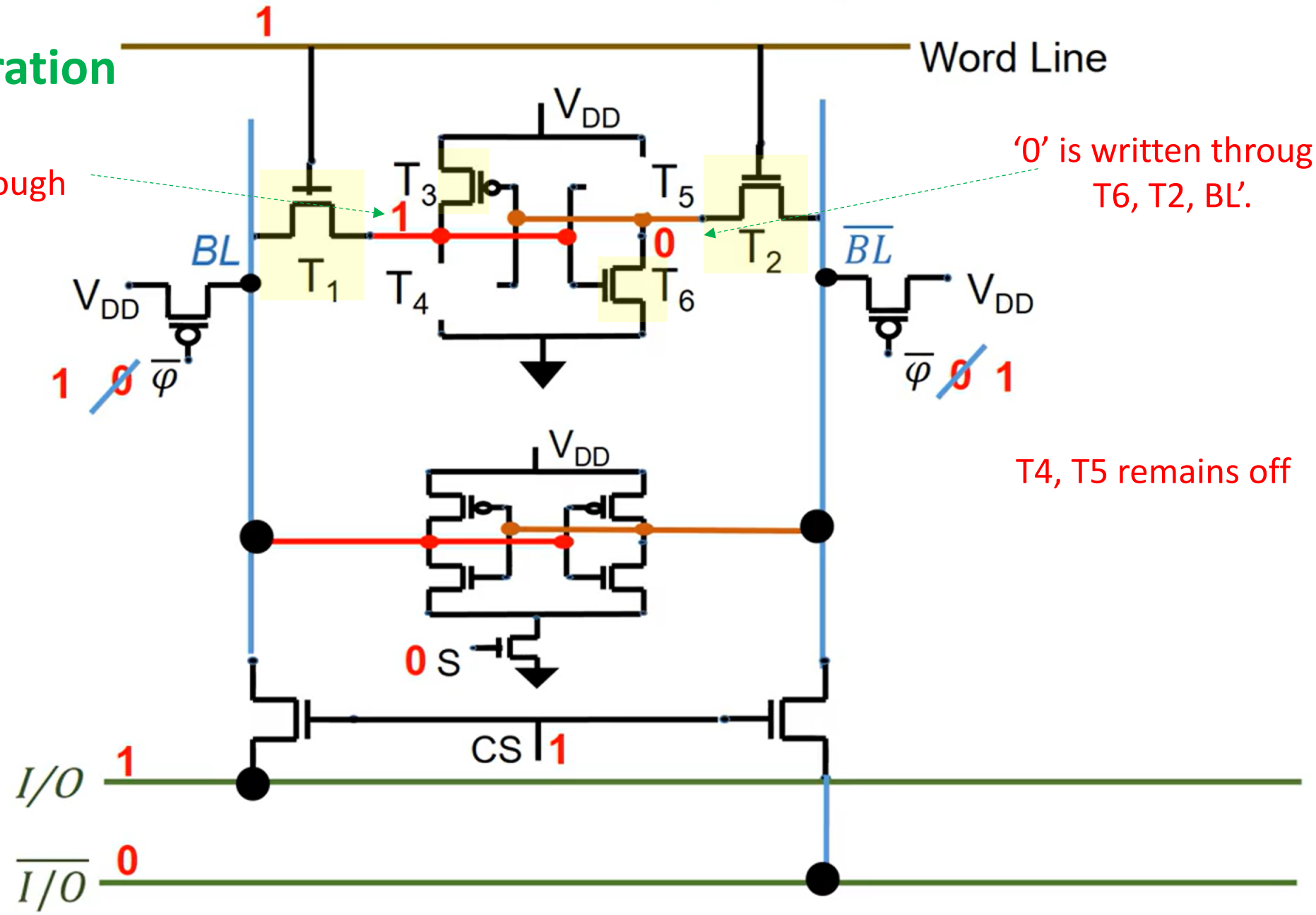


Random Access Memory (6T-SRAM)

Write operation

‘1’ is written through T3, T1, BL.

‘0’ is written through T6, T2, BL’.

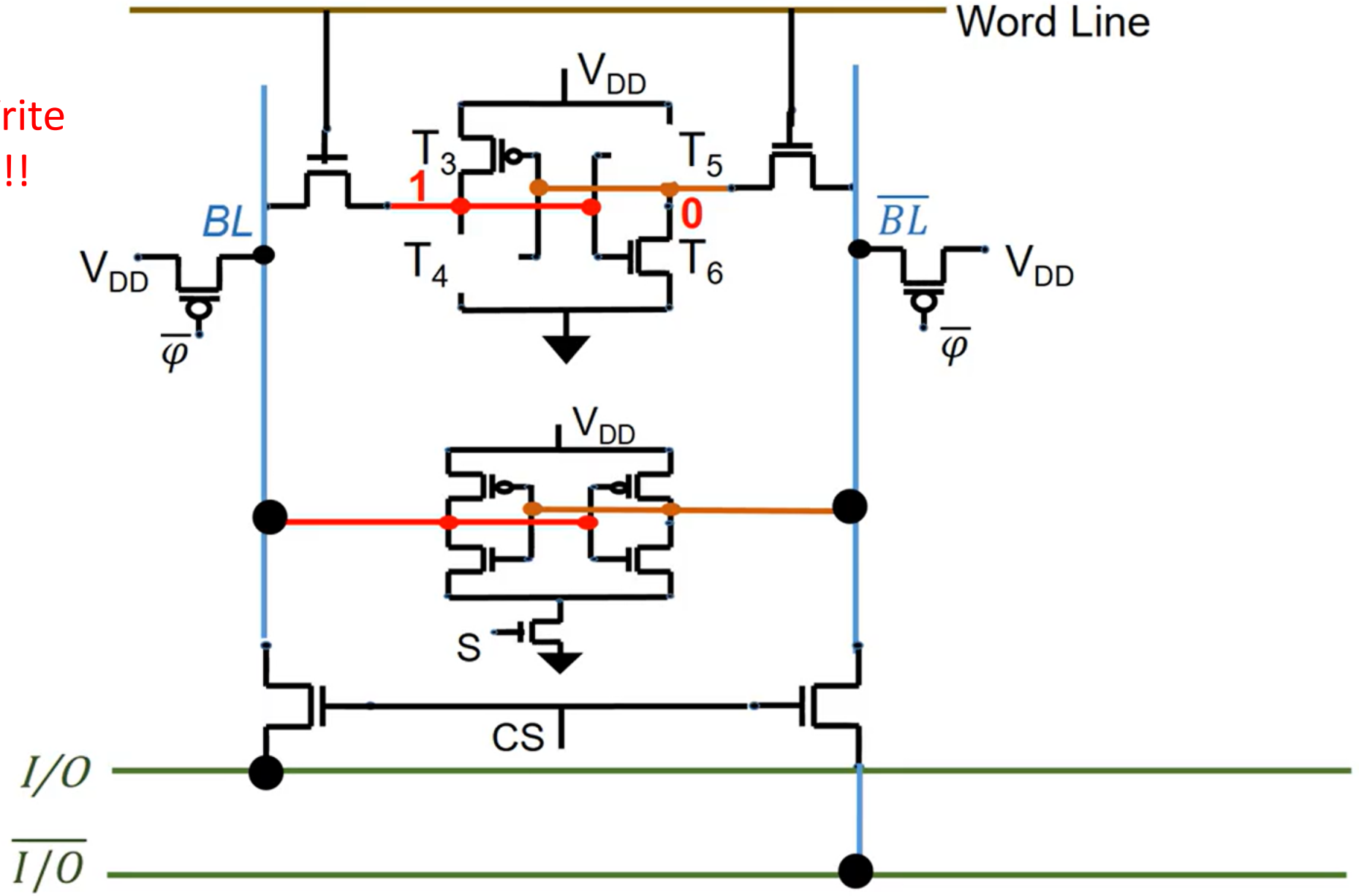


T4, T5 remains off

Write

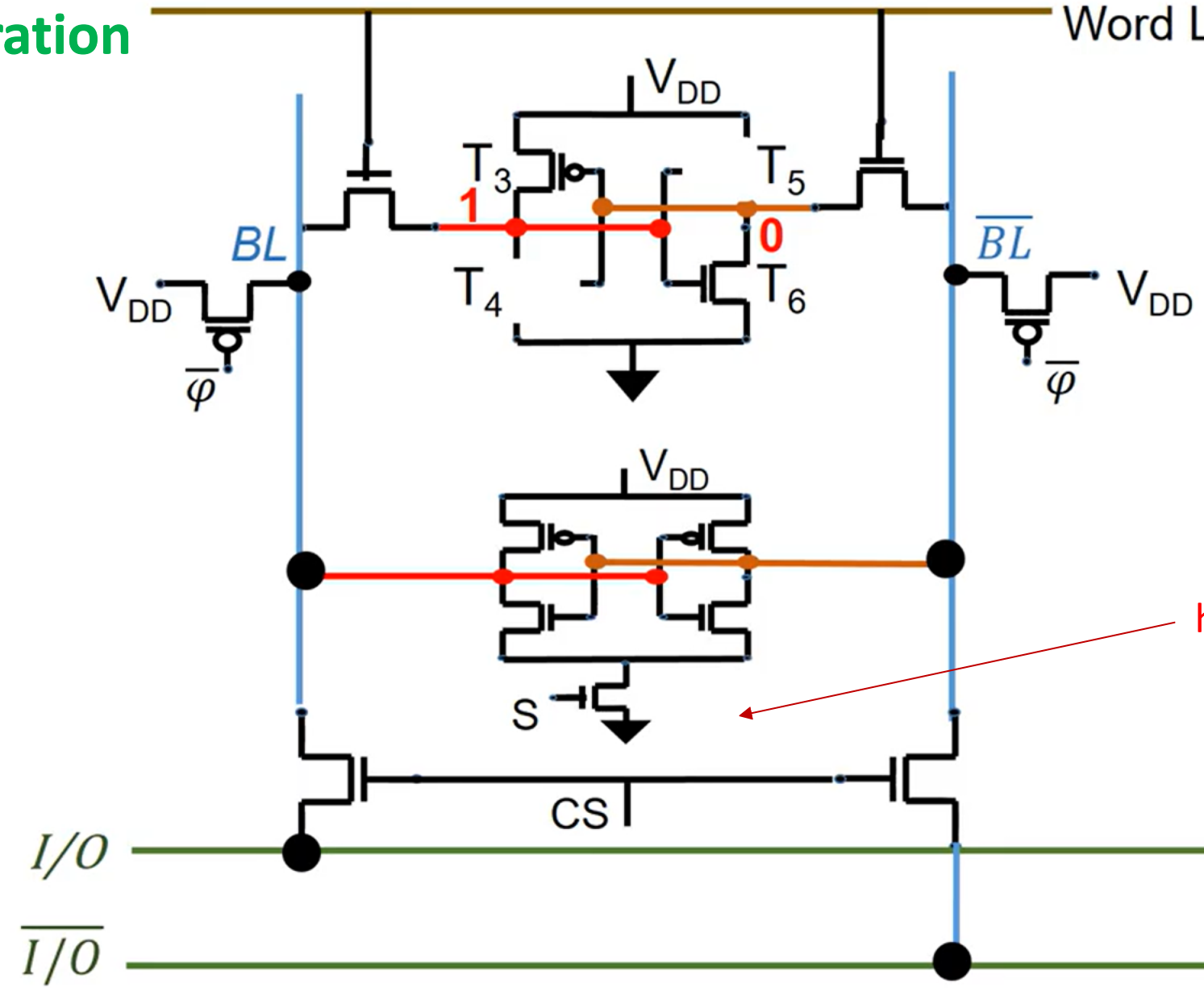
Random Access Memory (6T-SRAM)

After the Write operation!!



Random Access Memory (6T-SRAM)

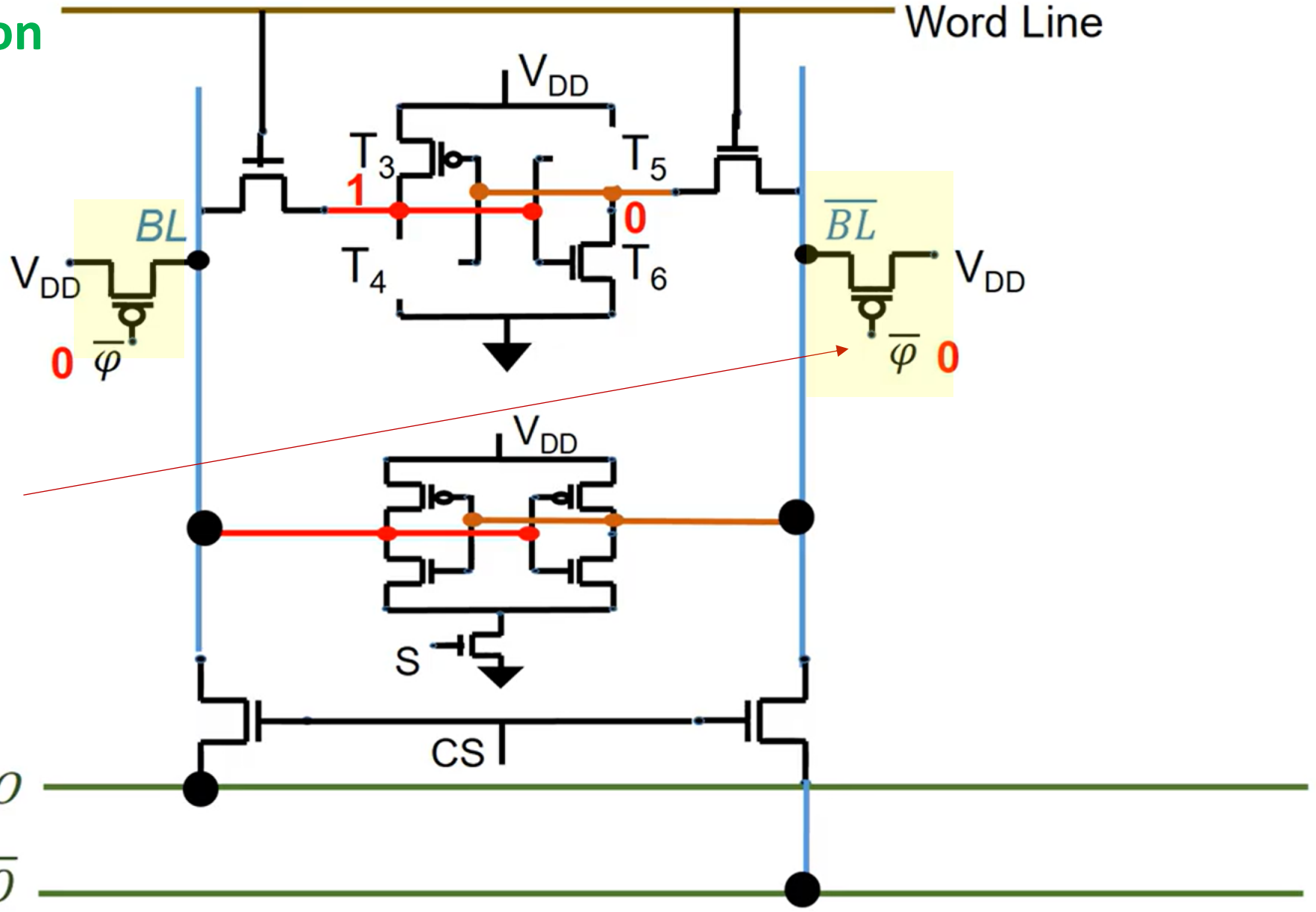
Read operation



Read operation, hence sense amplifier have to be made active soon…

Random Access Memory (6T-SRAM)

Read operation

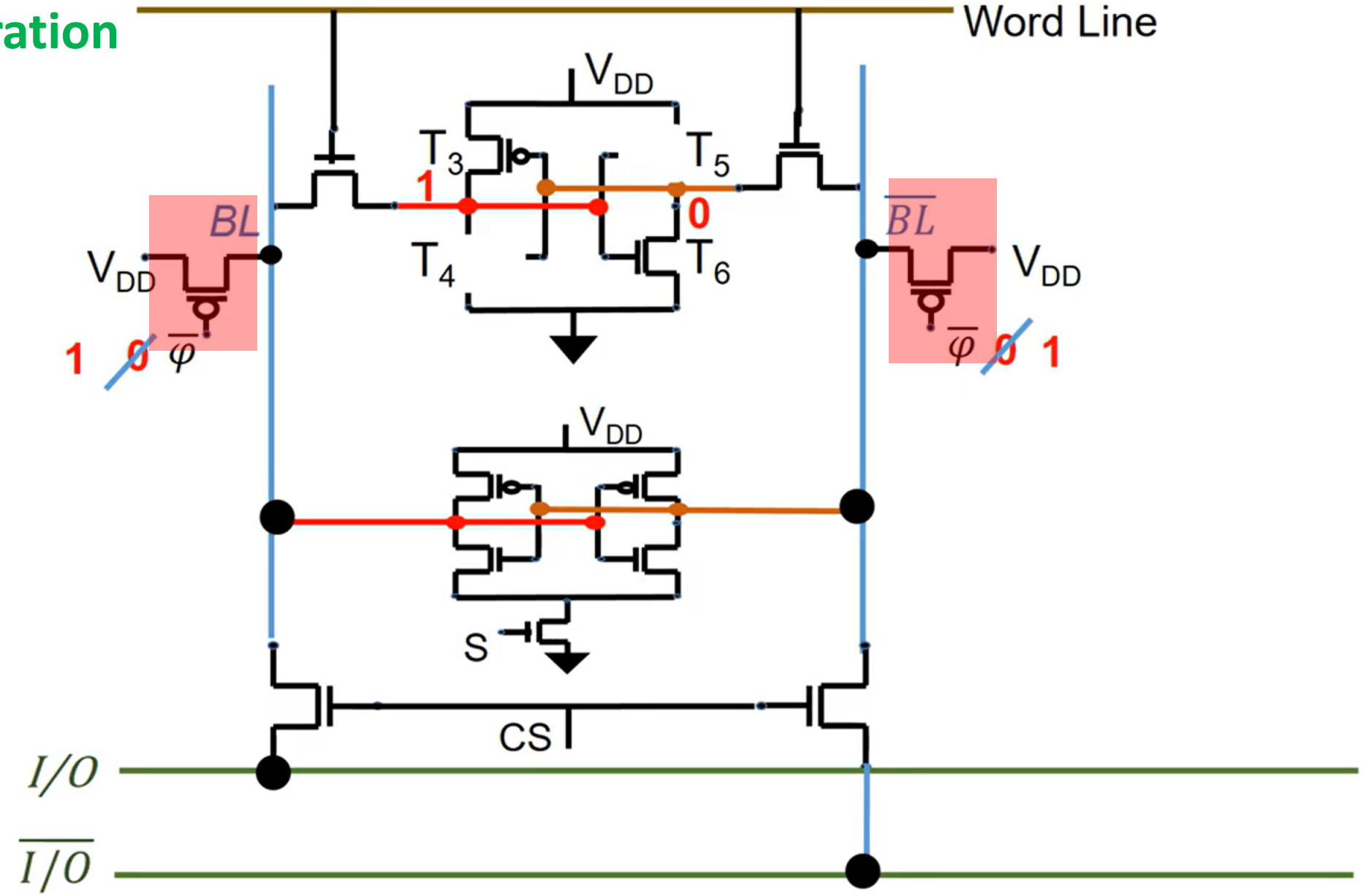


Precharging BL and BL'

Random Access Memory (6T-SRAM)

Read operation

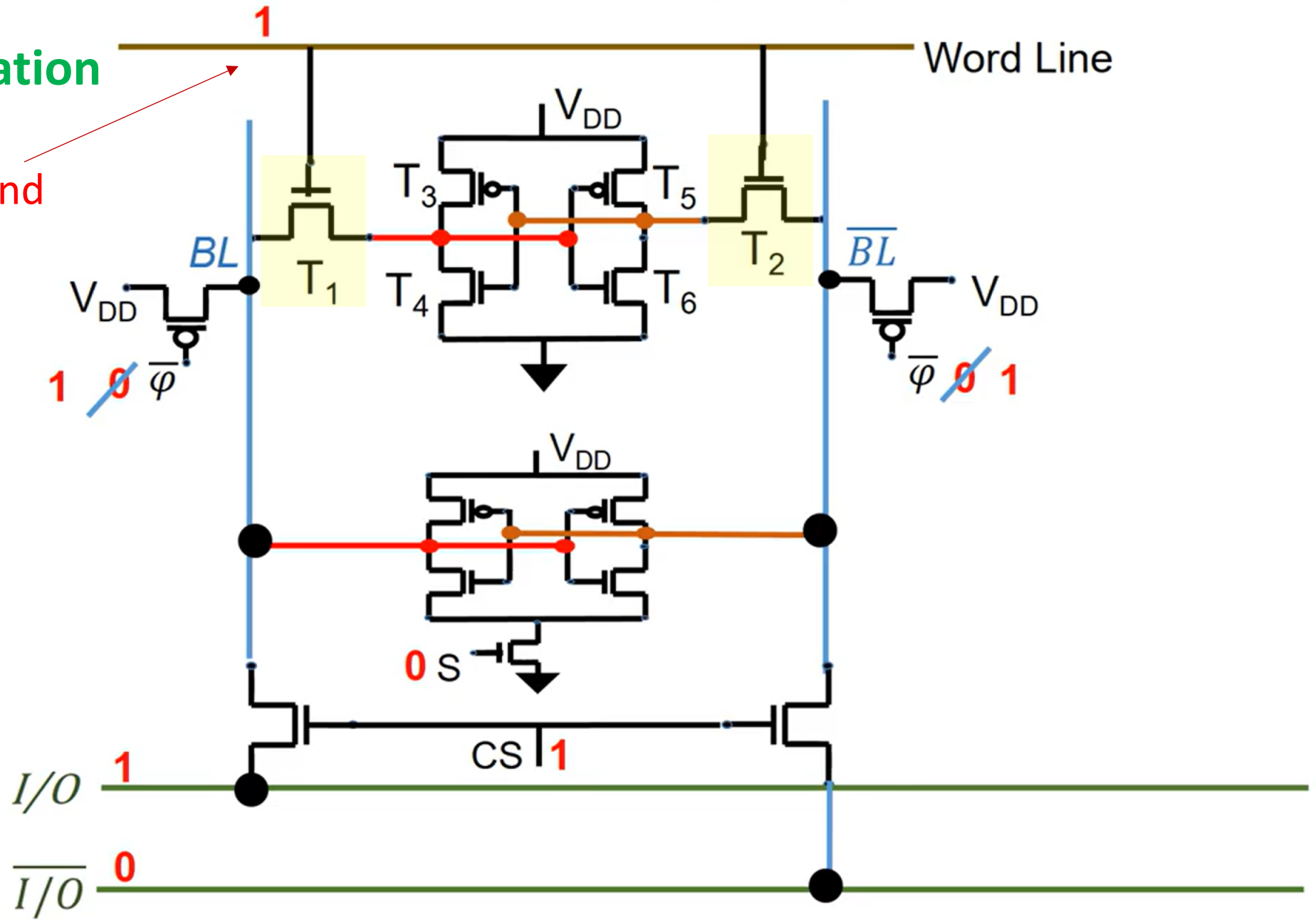
Precharging phase ends



Random Access Memory (6T-SRAM)

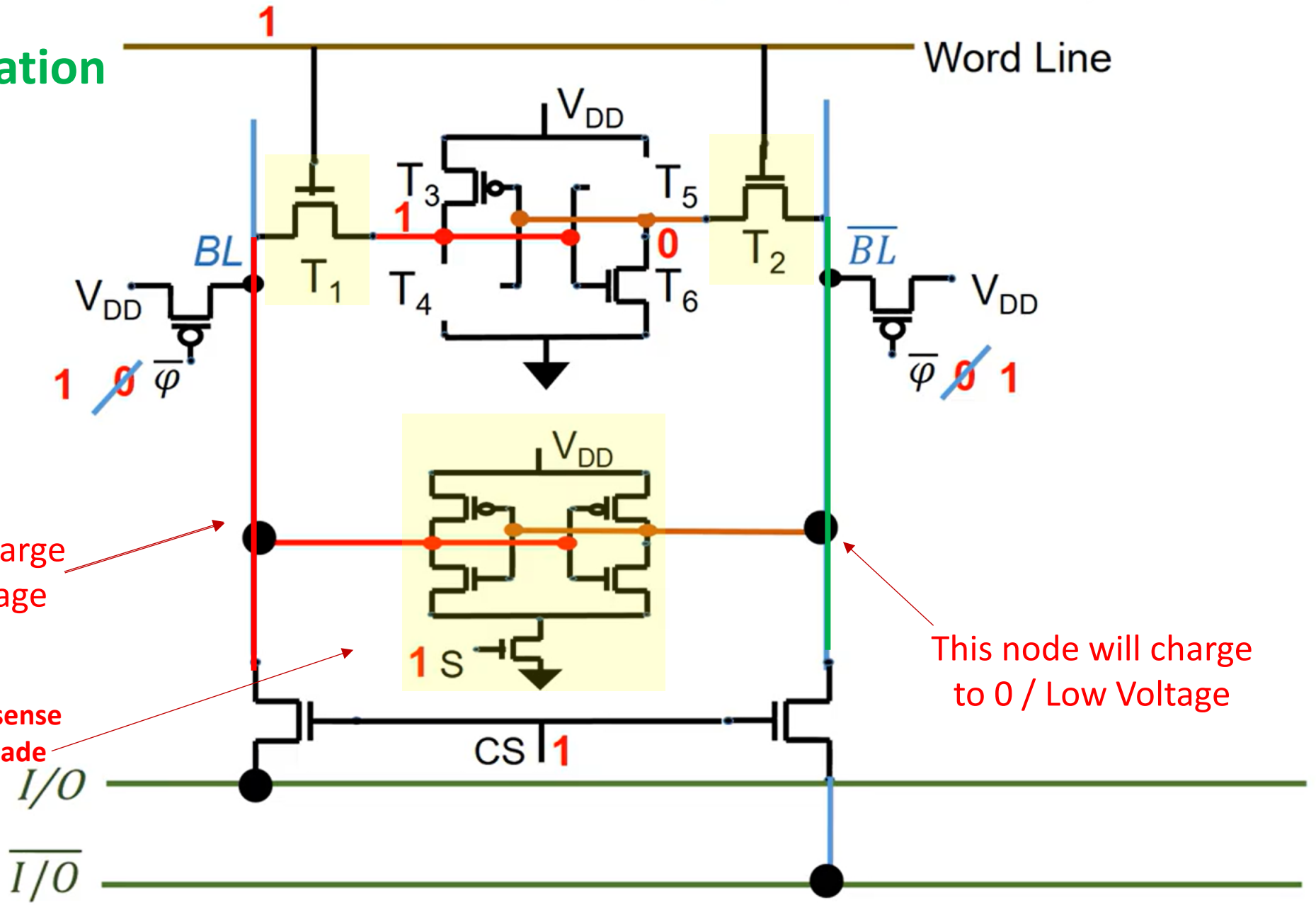
Read operation

Word line is activated, T1 and T2 are ON.



Random Access Memory (6T-SRAM)

Read operation



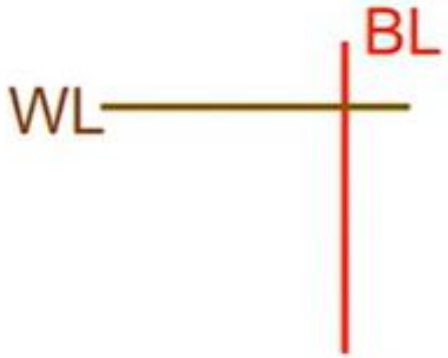
This node will charge to 1 / High Voltage

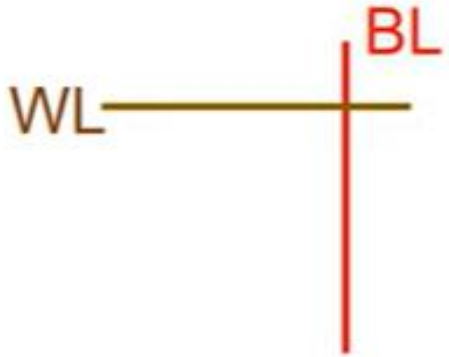
Read operation, hence sense amplifier have to be made active

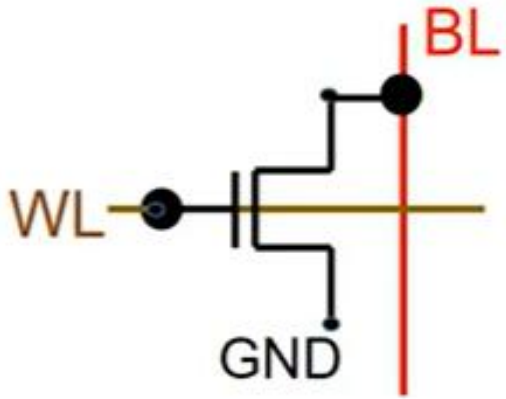
This node will charge to 0 / Low Voltage

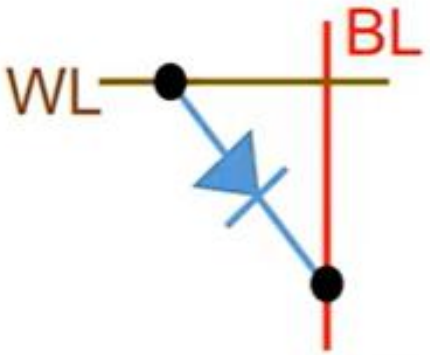
Read Only Memory (ROM)

Read-Only Memory Cells

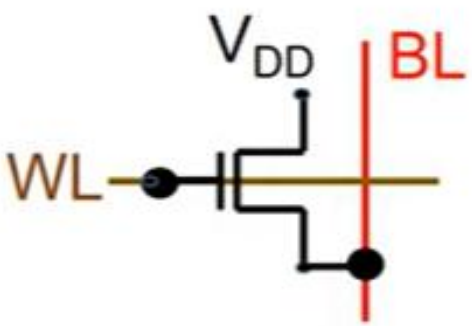








Diode ROM



MOS ROM-1

MOS ROM-2

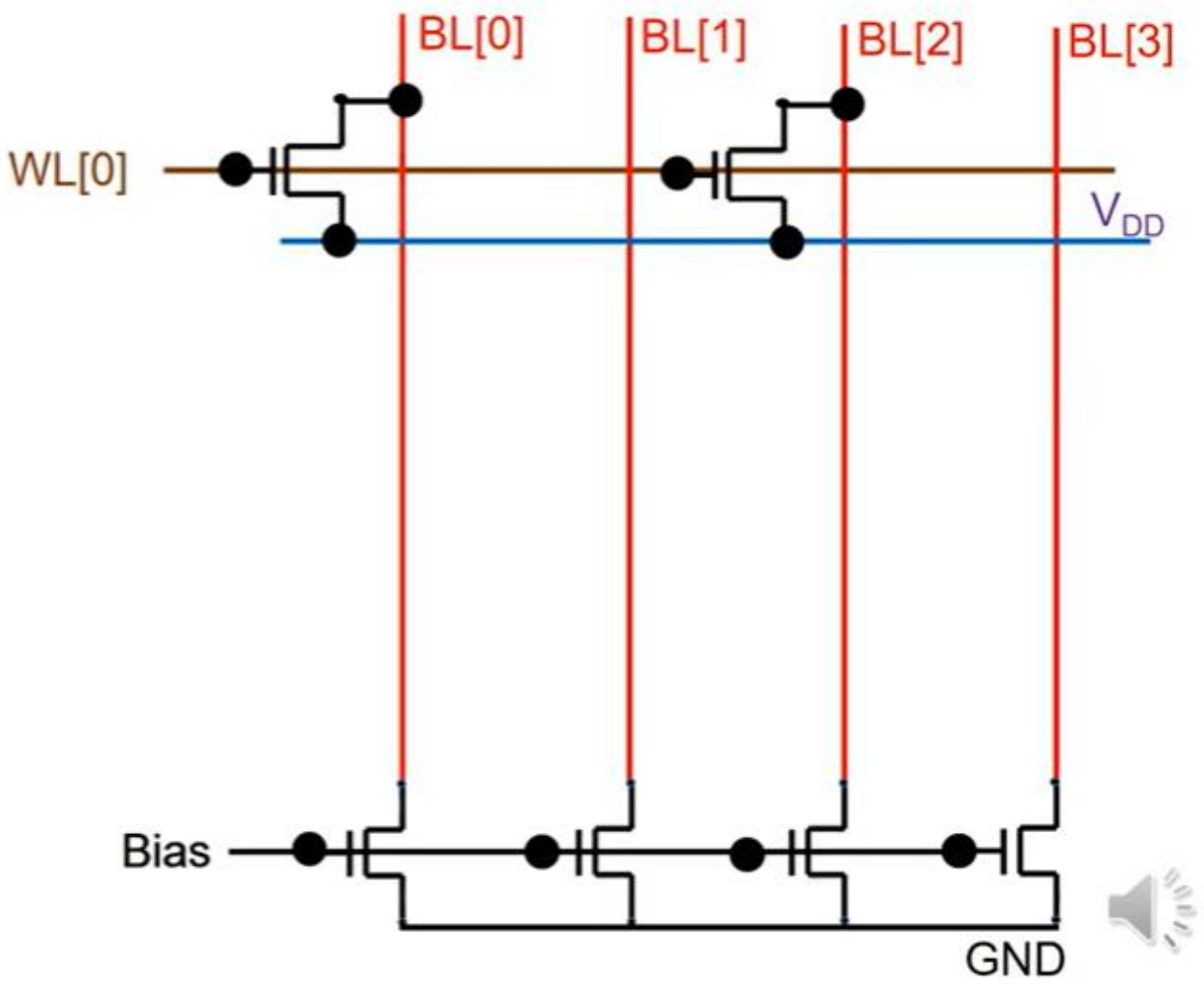
Three types of ROM

1. OR ROM
2. NOR ROM
3. NAND ROM

MOS OR ROM

	BL[0]	BL[1]	BL[2]	BL[3]
WL[0]=	1	0	1	0
WL[1]=	0	1	1	0
WL[2]=	0	0	0	1
WL[3]=	0	1	1	0

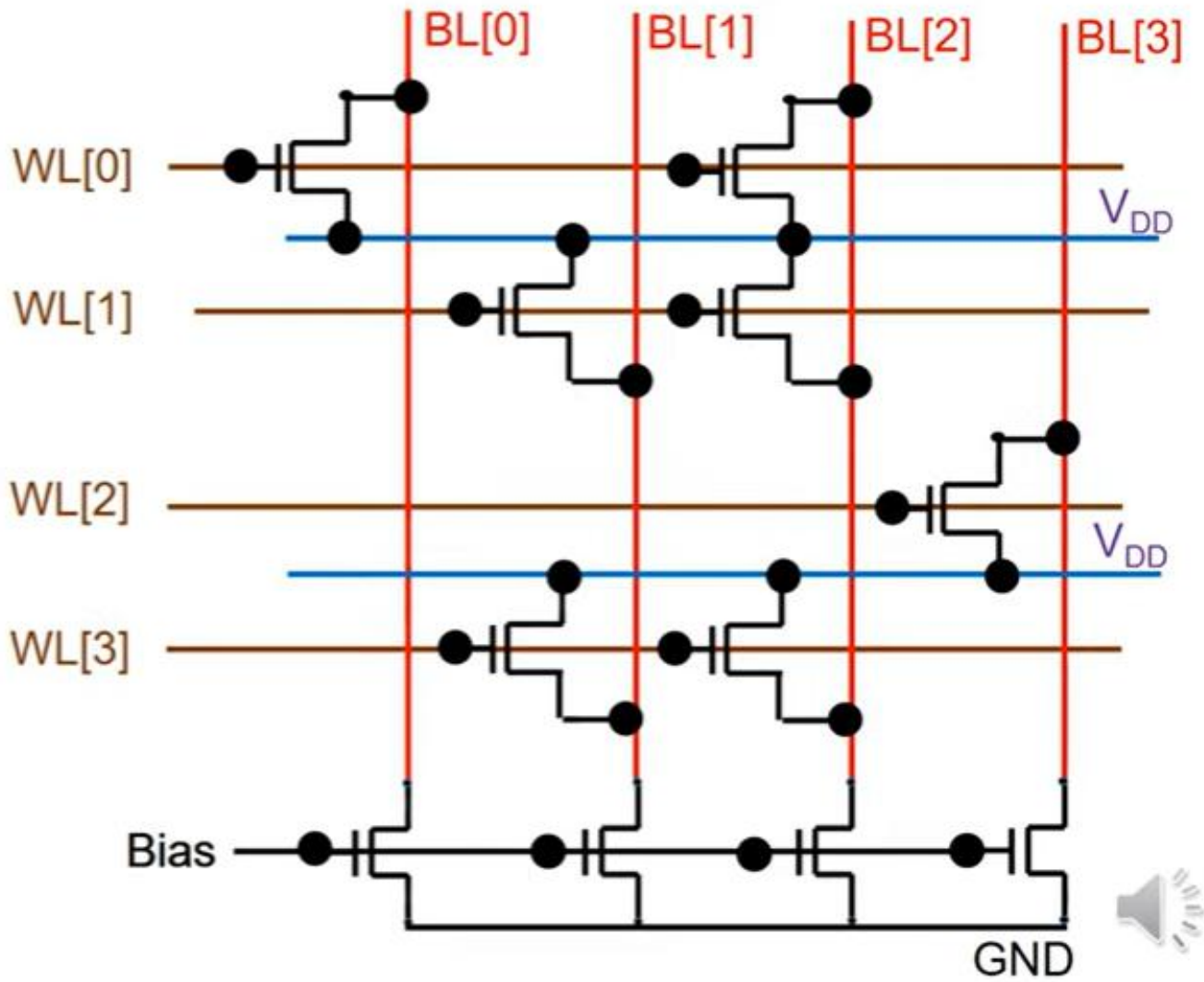
Active high decoder



MOS OR ROM

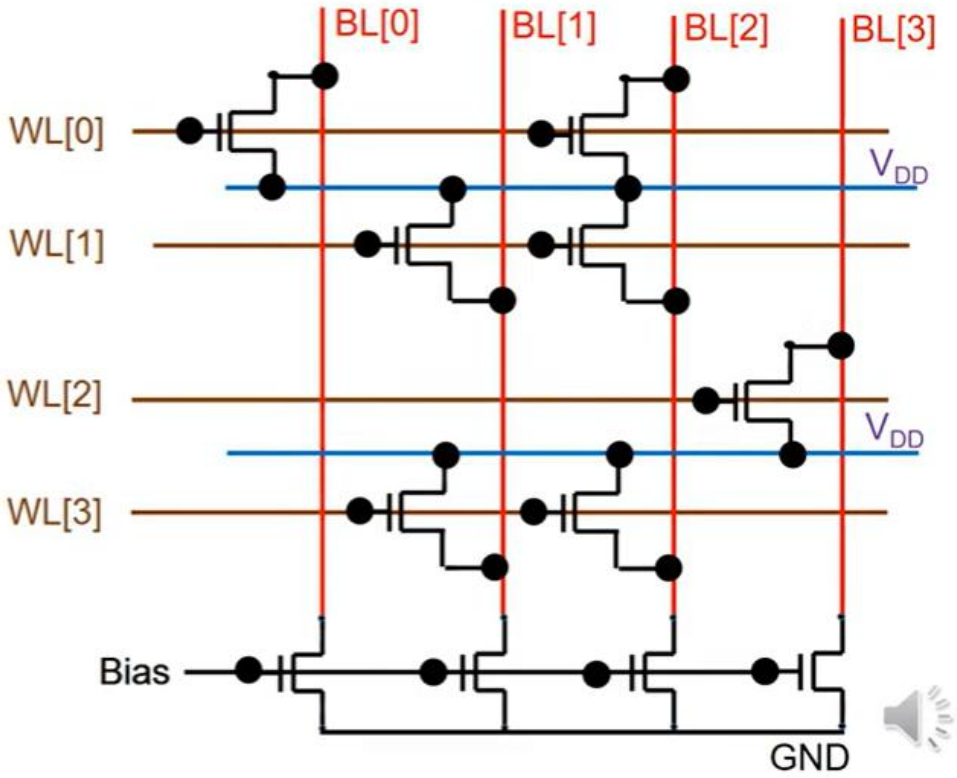
	BL[0]	BL[1]	BL[2]	BL[3]
WL[0]=	1	0	1	0
WL[1]=	0	1	1	0
WL[2]=	0	0	0	1
WL[3]=	0	1	1	0

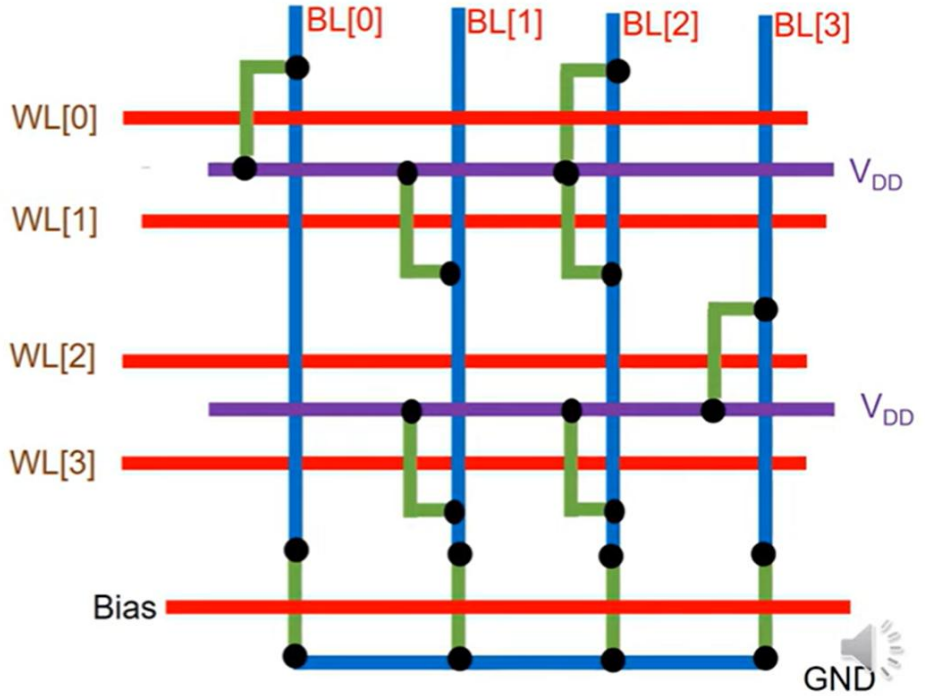
Active high decoder



MOS OR ROM

=



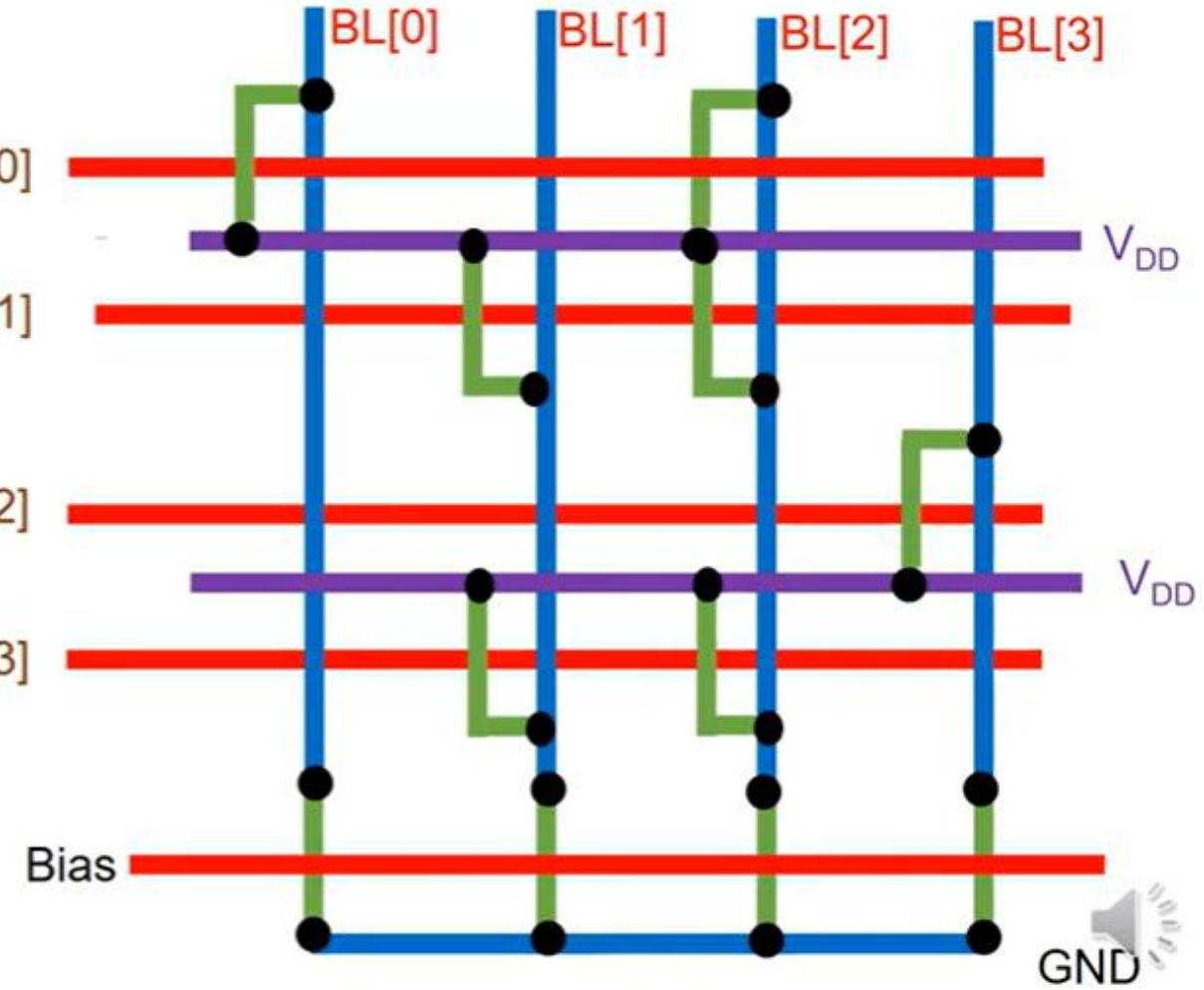


WL[0]= 1 0 1 0
WL[1]= 0 1 1 0
WL[2]= 0 0 0 1
WL[3]= 0 1 1 0

Active high decoder

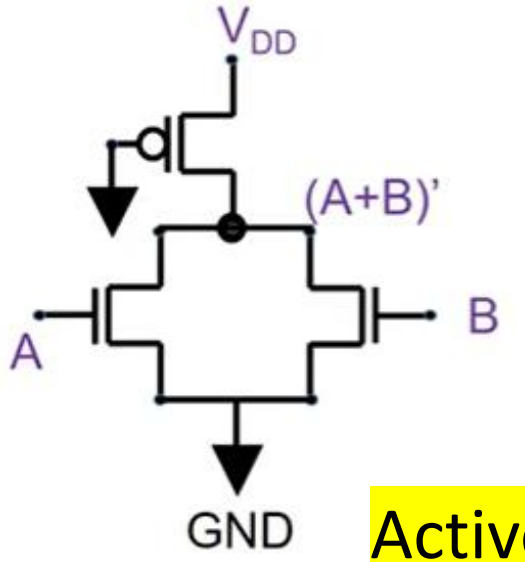
MOS OR ROM

	BL[0]	BL[1]	BL[2]	BL[3]
WL[0]=	1	0	1	0
WL[1]=	0	1	1	0
WL[2]=	0	0	0	1
WL[3]=	0	1	1	0

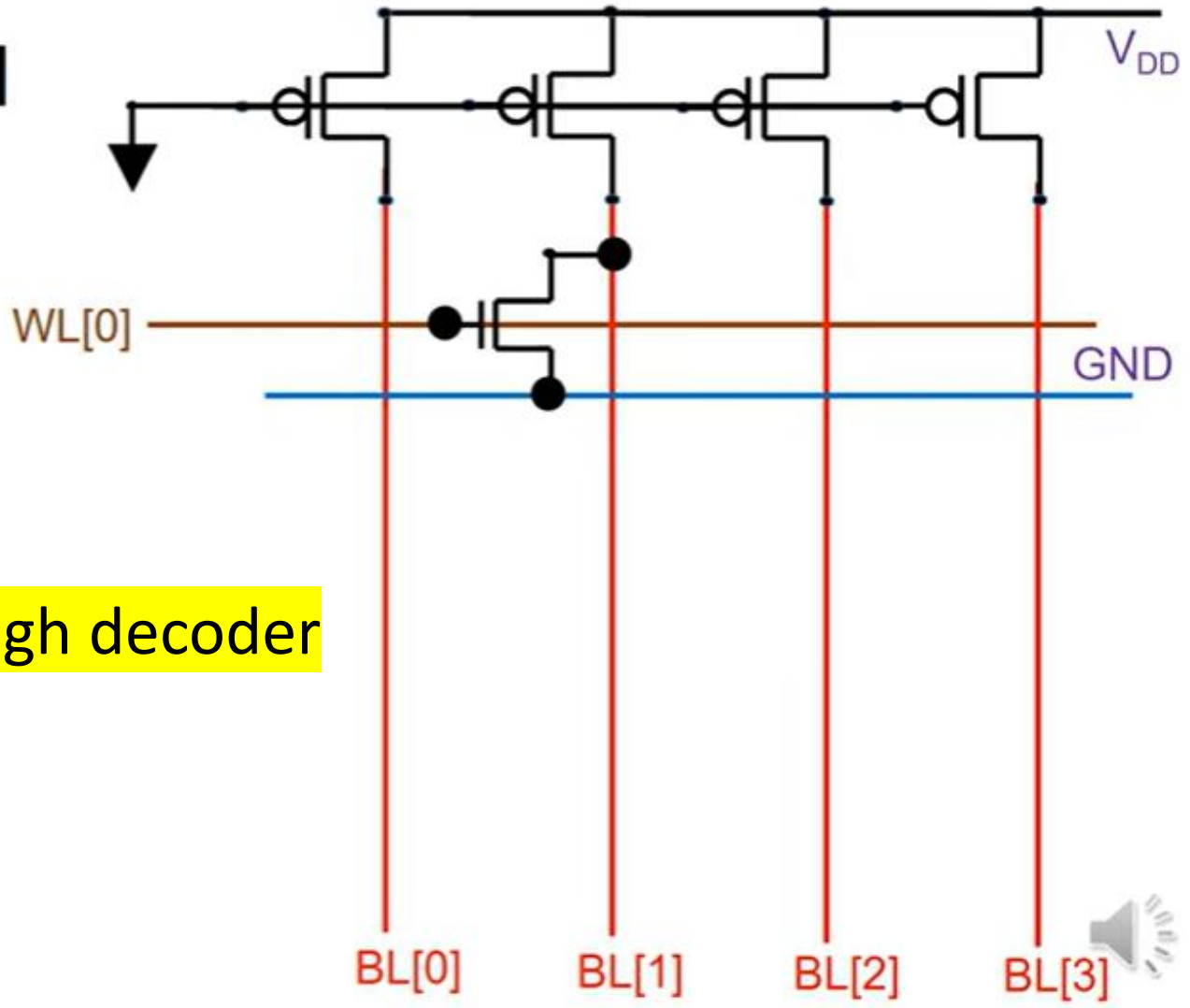


Active high decoder

MOS NOR ROM

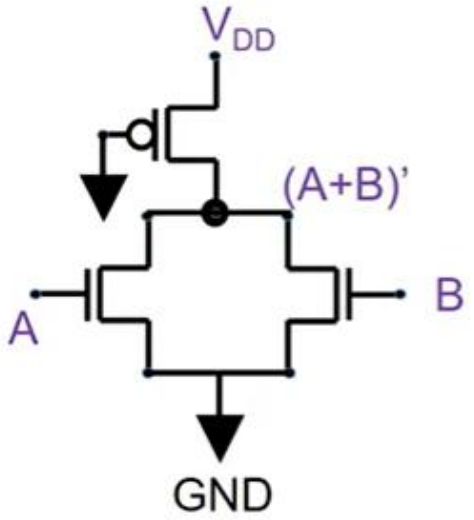


Active high decoder



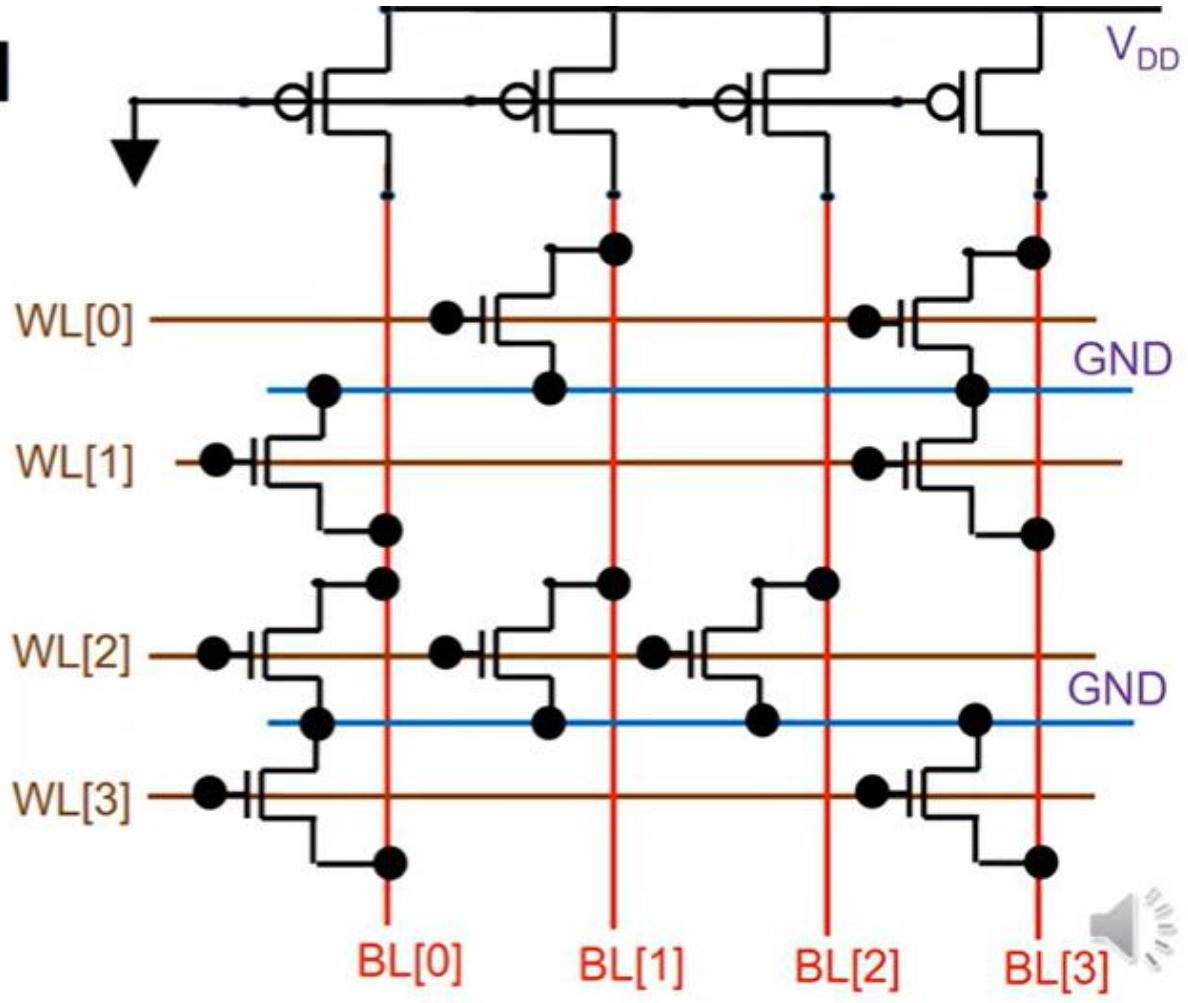
WL[0]= 1 0 1 0
WL[1]= 0 1 1 0
WL[2]= 0 0 0 1
WL[3]= 0 1 1 0

MOS NOR ROM



WL[0]= 1 0 1 0
WL[1]= 0 1 1 0
WL[2]= 0 0 0 1
WL[3]= 0 1 1 0

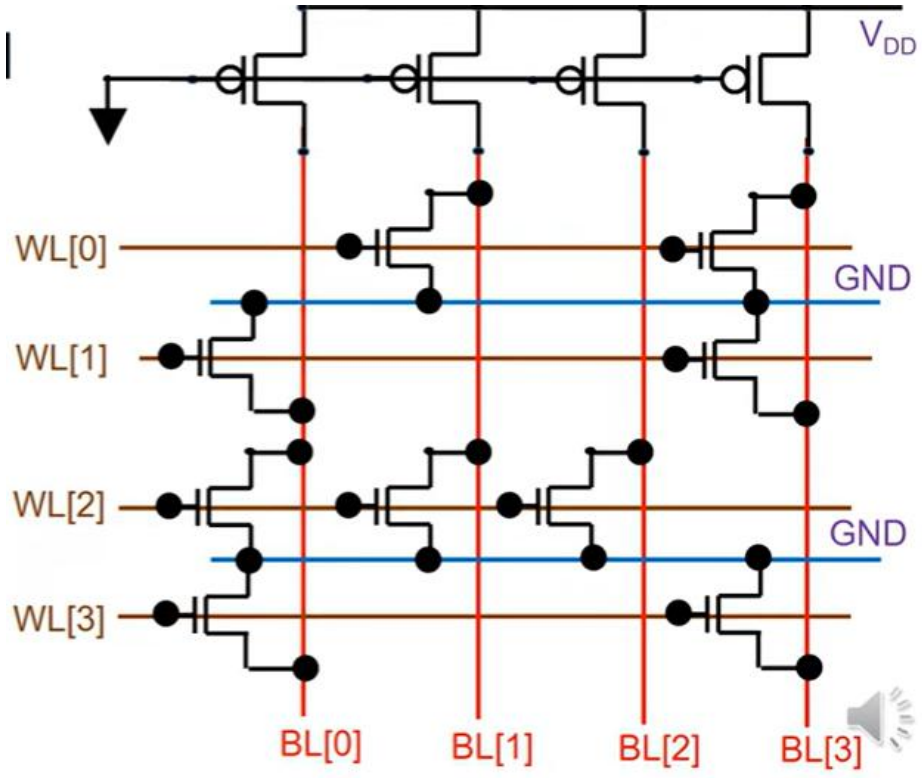
=

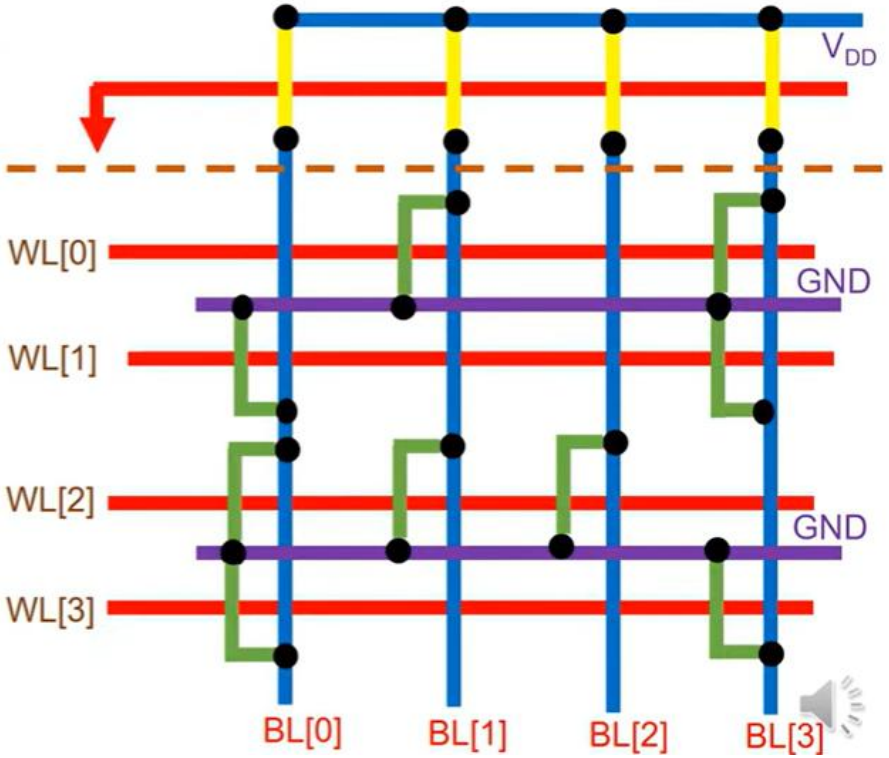


Active high decoder

MOS NOR ROM

WL[0]= 1 0 1 0
WL[1]= 0 1 1 0
WL[2]= 0 0 0 1
WL[3]= 0 1 1 0





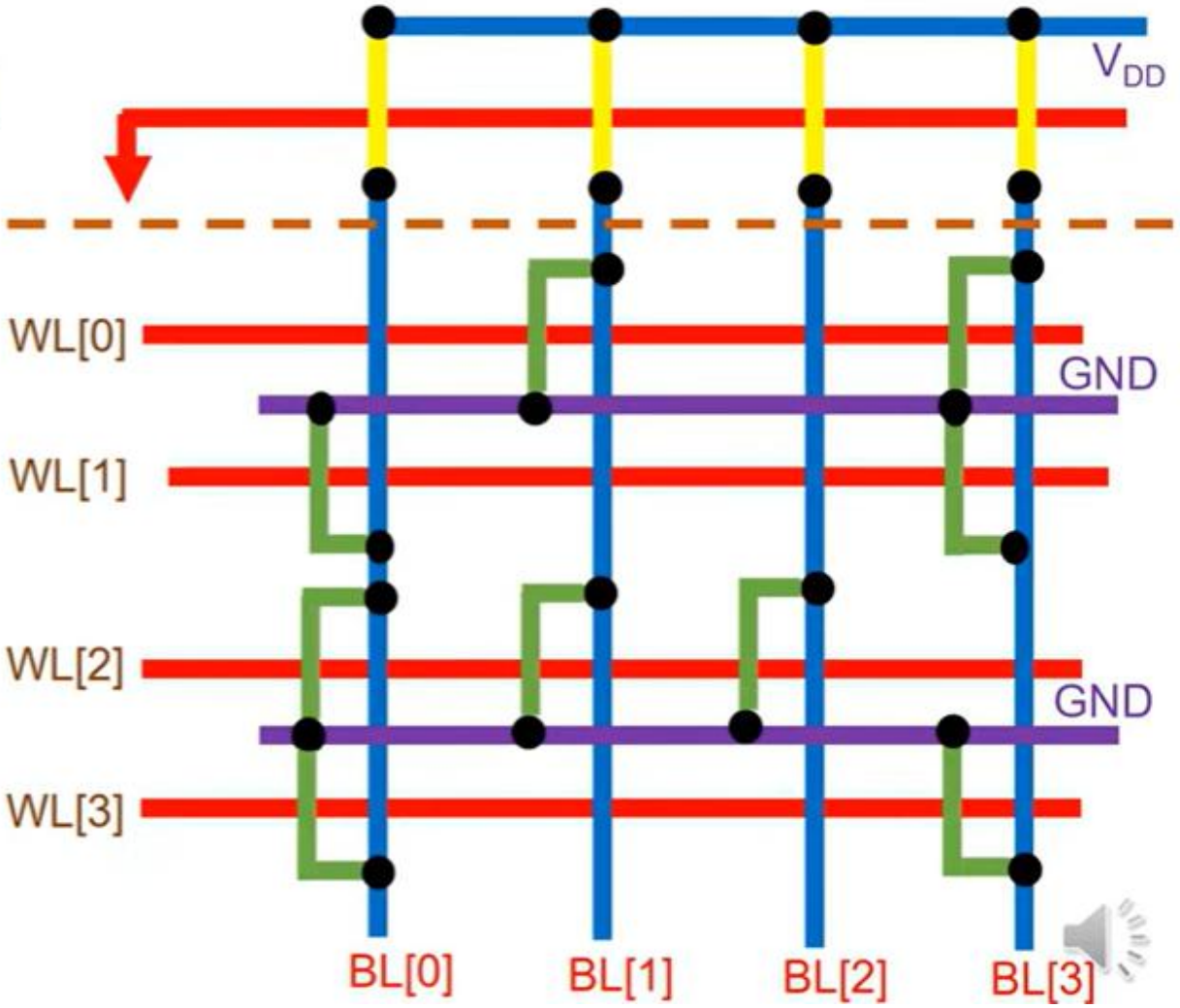
Active high decoder

=

MOS NOR ROM

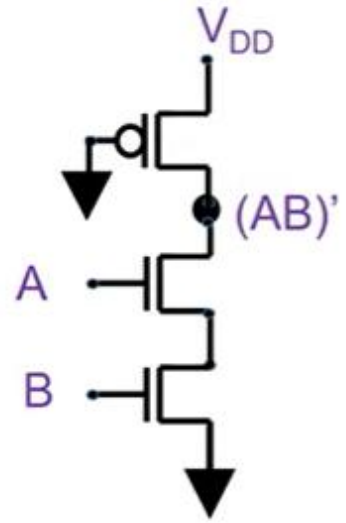
WL[0]= 1 0 1 0
WL[1]= 0 1 1 0
WL[2]= 0 0 0 1
WL[3]= 0 1 1 0

Active high decoder

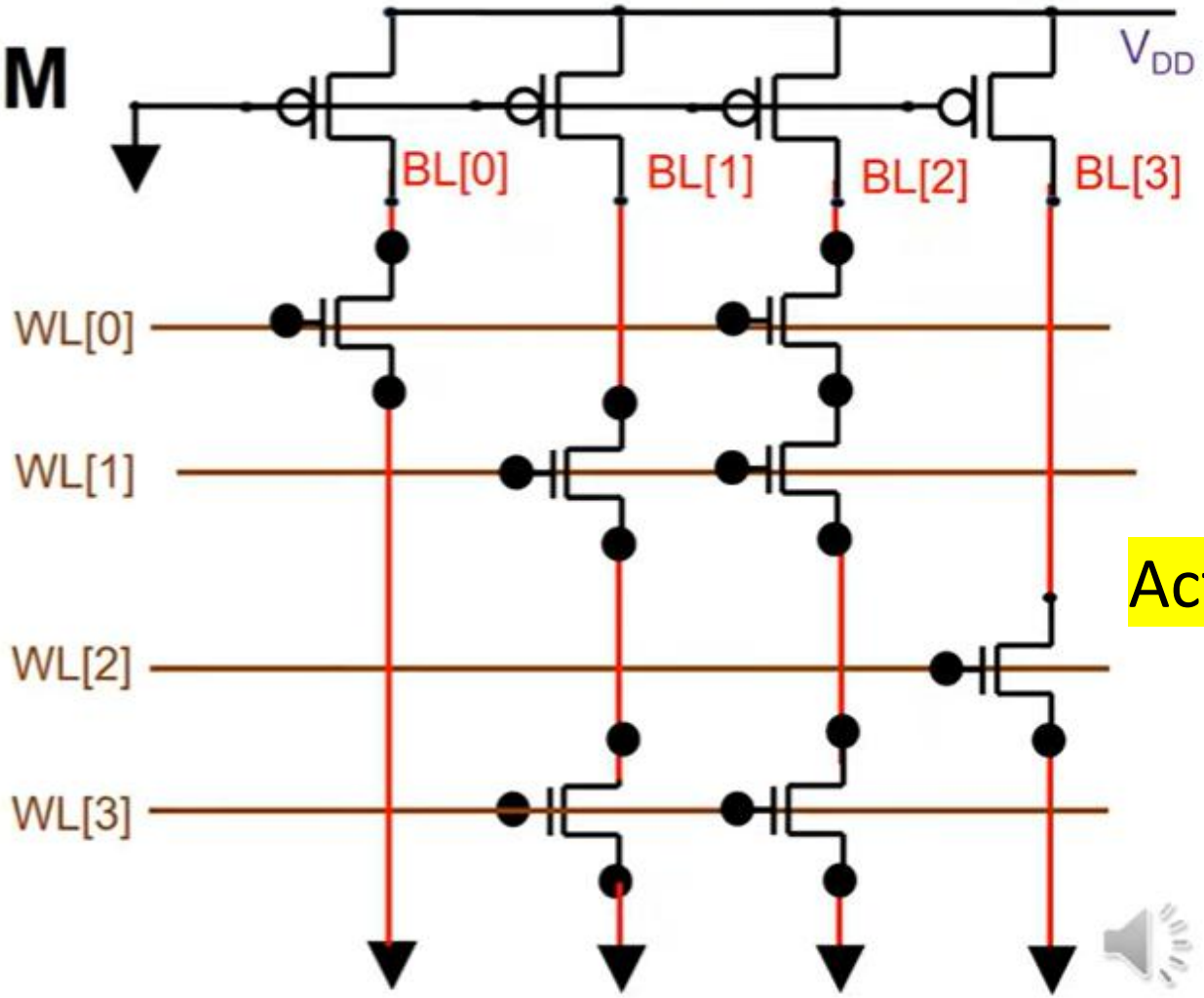


=

MOS NAND ROM

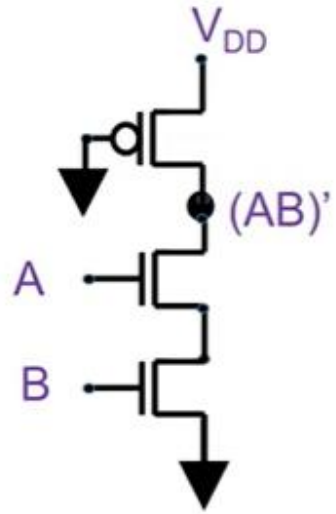


WL[0]= 1 0 1 0
WL[1]= 0 1 1 0
WL[2]= 0 0 0 1
WL[3]= 0 1 1 0

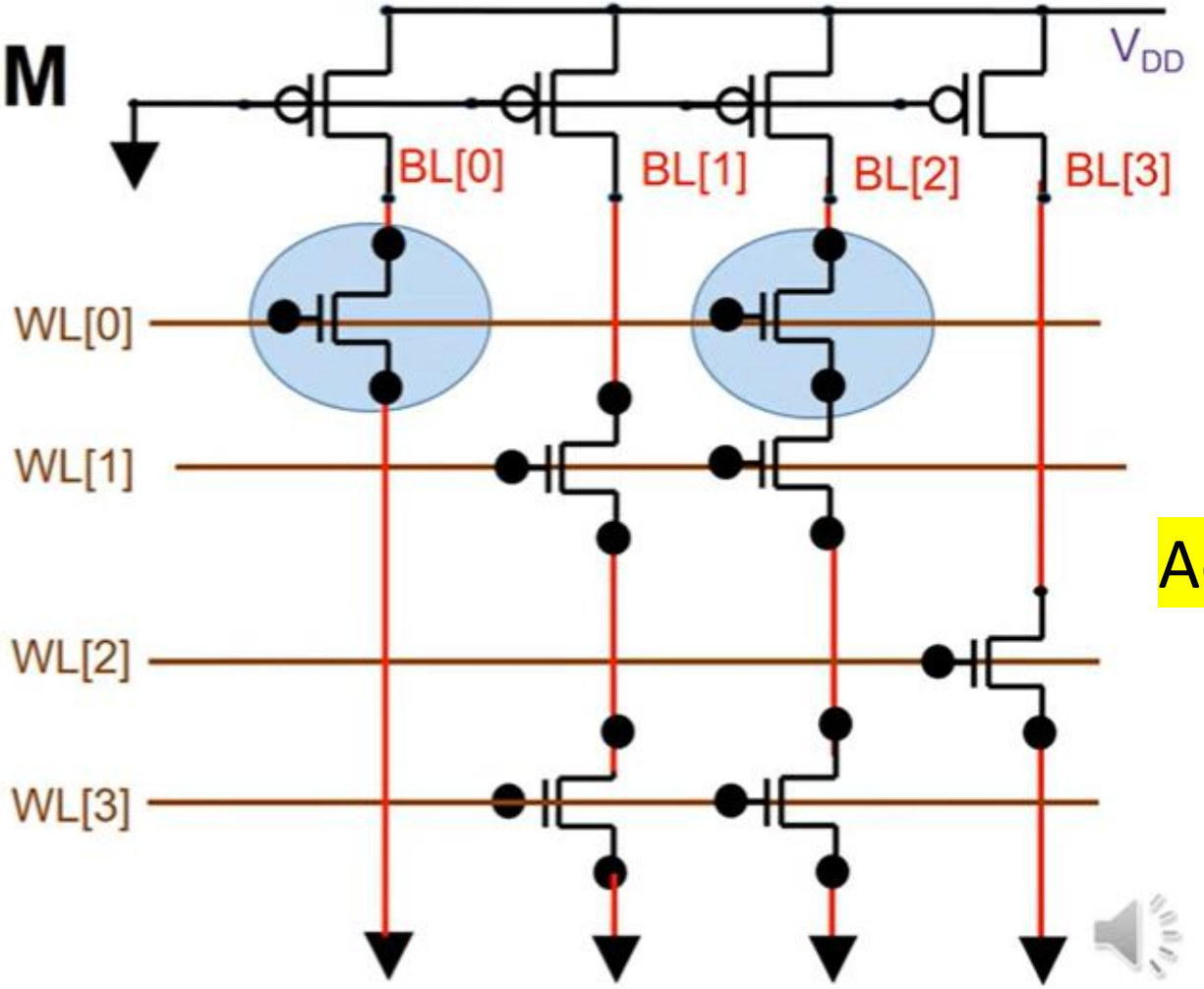


Active Low decoder

MOS NAND ROM



WL[0]= 1 0 1 0
WL[1]= 0 1 1 0
WL[2]= 0 0 0 1
WL[3]= 0 1 1 0

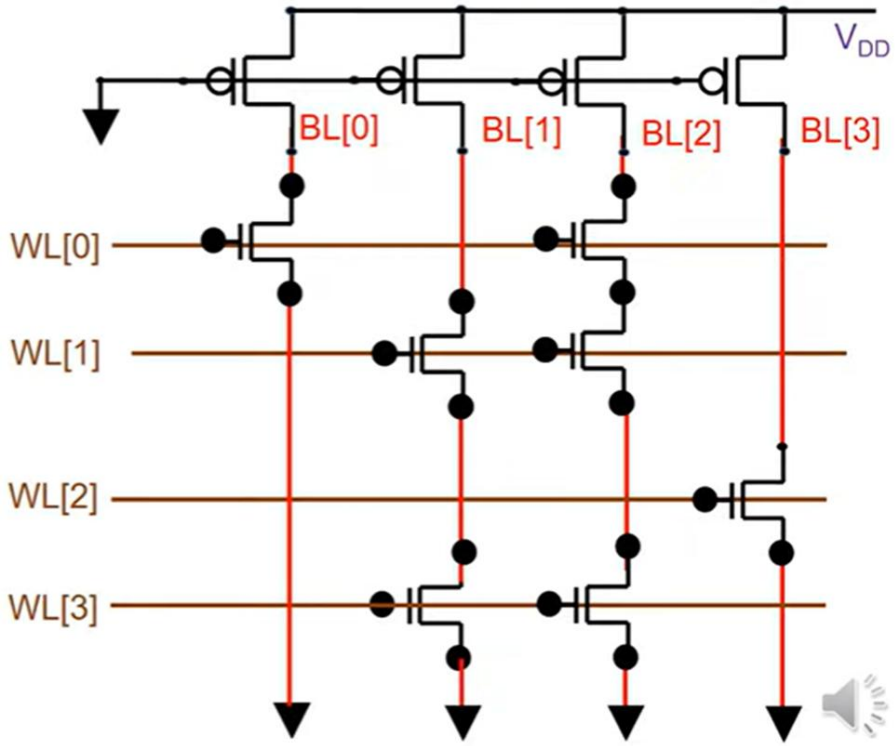


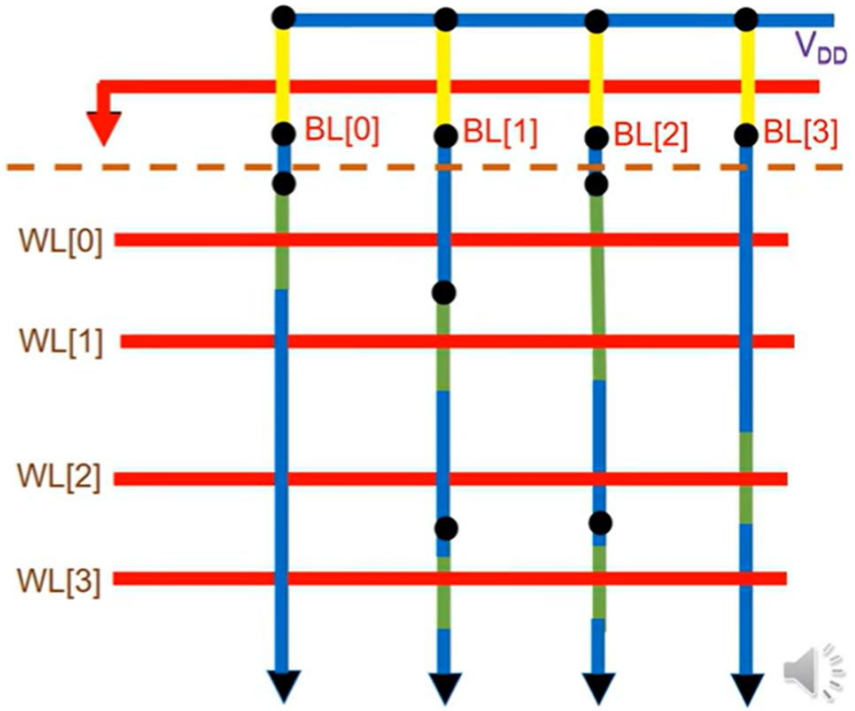
Active Low decoder

MOS NAND ROM

=

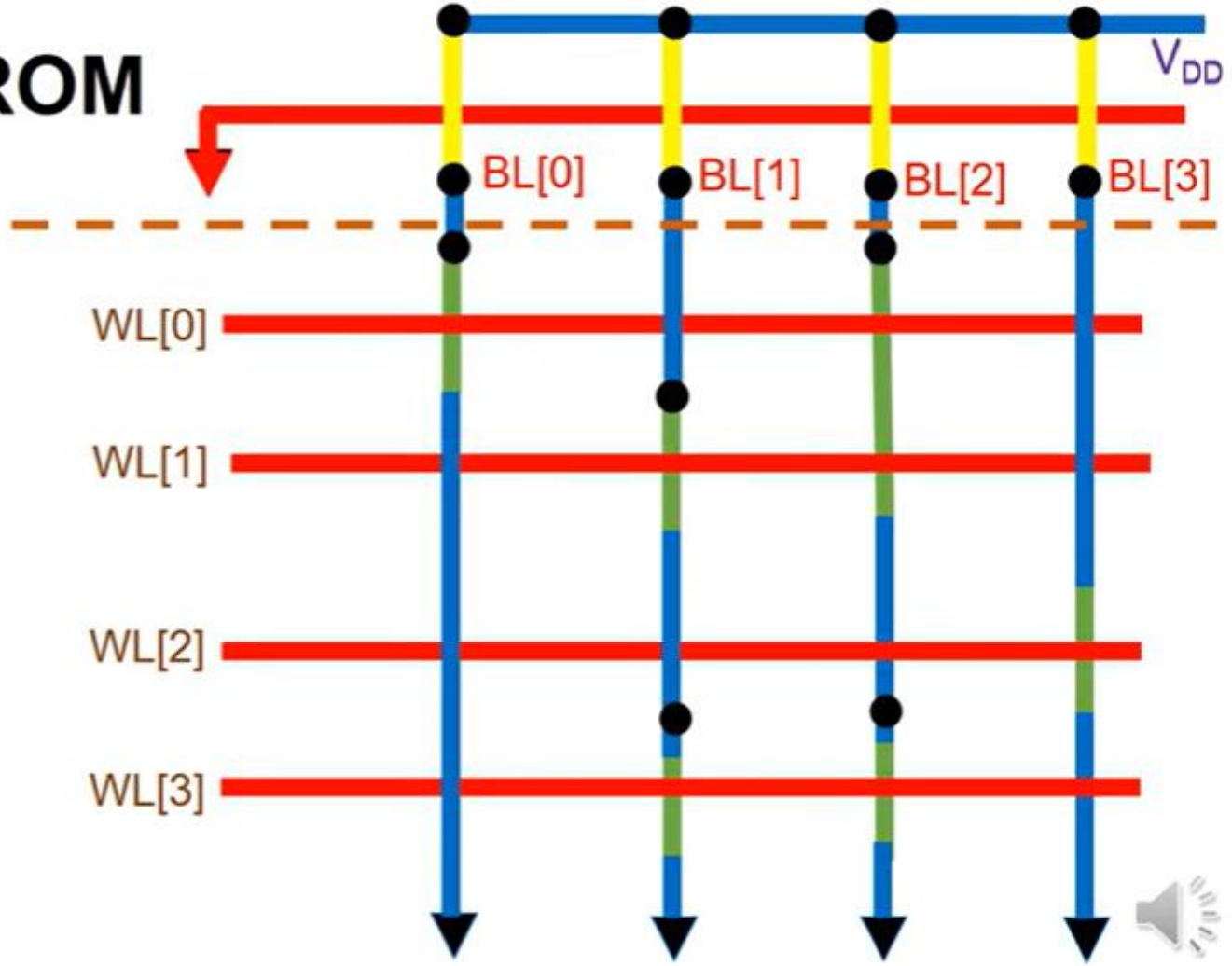
WL[0]= 1 0 1 0
WL[1]= 0 1 1 0
WL[2]= 0 0 0 1
WL[3]= 0 1 1 0





Active Low decoder

MOS NAND ROM



WL[0]= 1 0 1 0
WL[1]= 0 1 1 0
WL[2]= 0 0 0 1
WL[3]= 0 1 1 0

Active Low decoder

=

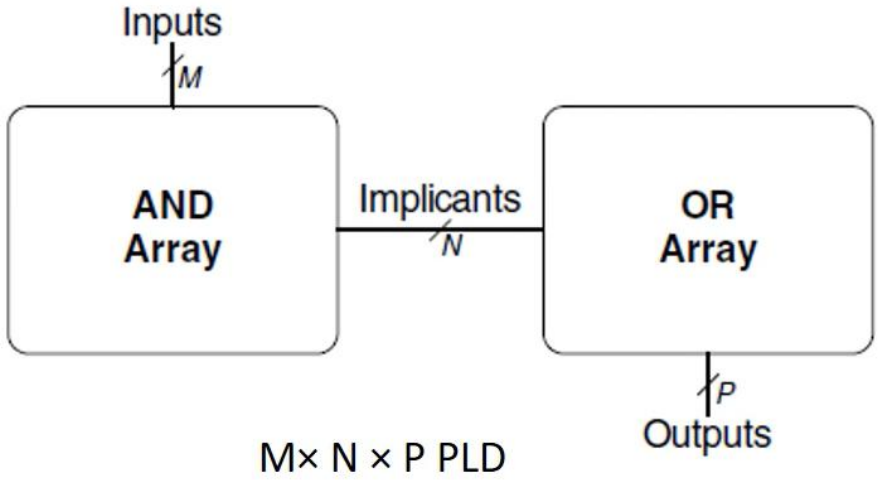
Q.1:Implement following 4-bit words using NOR and NAND ROM structure. W[0]: 0101; W[1]: 0011; W[2]: 1001; W[3]: 0110.

VLSI Design: Subsystem Design

- PLD
- Bus arbitration logic
- Parity Generator
- ALU
- Crossbar switch, shifter
- Barrel Shifter

Programmable Logic Device (PLD)

Definition: A *Programmable Logic Device* (PLD) is a chip that is manufactured with a programmable configuration, enabling it to serve in many arbitrary applications.



M× N × P PLD

Programmable Read Only Memory (*PROMs*) implement two-level **combinational logic** in sum-of-products (SOP) form. To implement the sequential logic we need additional memory element i.e. Flip-Flops.

Programmable Logic Devices(PLD)

Types of the AND–OR structured programmable logic devices are

1. Programmable read-only memory (PROM)
2. Programmable logic array (PLA)
3. Programmable array logic (PAL)

PLD	AND Plane	OR Plane
PROM	Fixed	Programmable
PLA	Programmable	Programmable
PAL	Programmable	Fixed

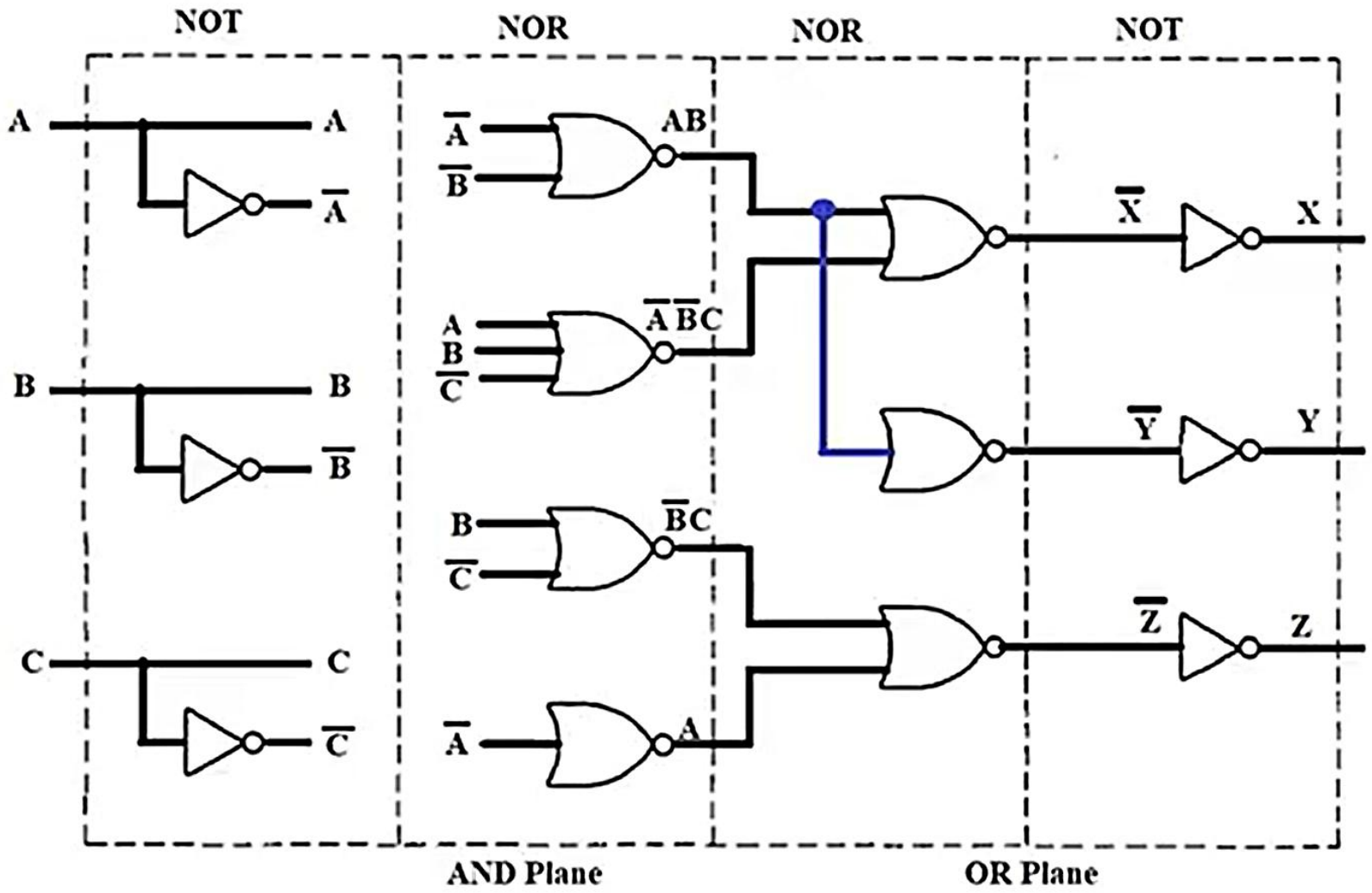
Programmable Logic Array (PLA)

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

$$Y = AB$$

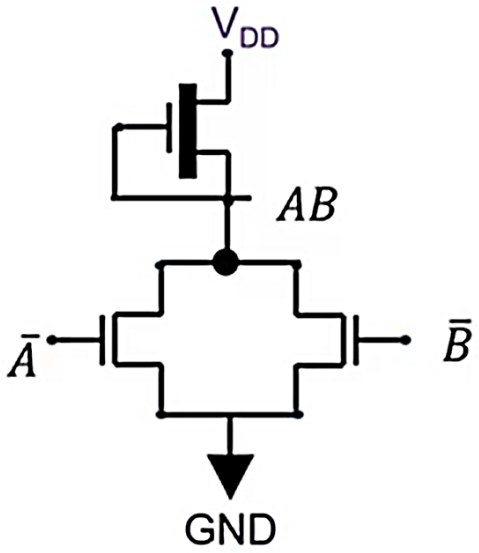
$$Z = A + \bar{B}C$$

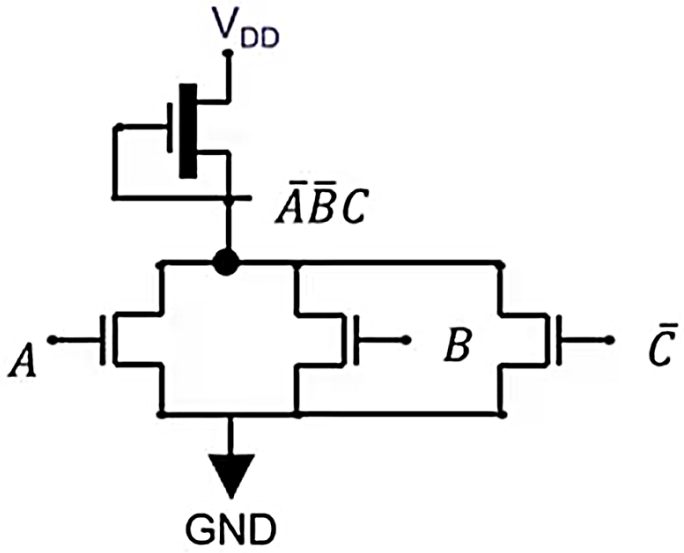
1. **nMOS PLA**
2. **CMOS PLA**



1. nMOS PLA

=

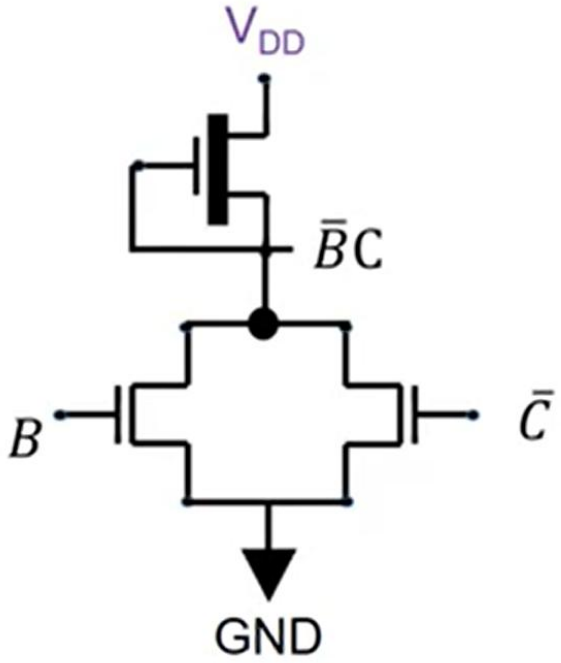


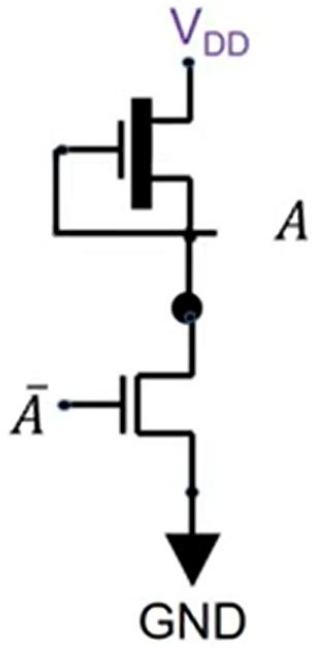


$$X = AB + \bar{A}\bar{B}C$$

$$Y = AB$$

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



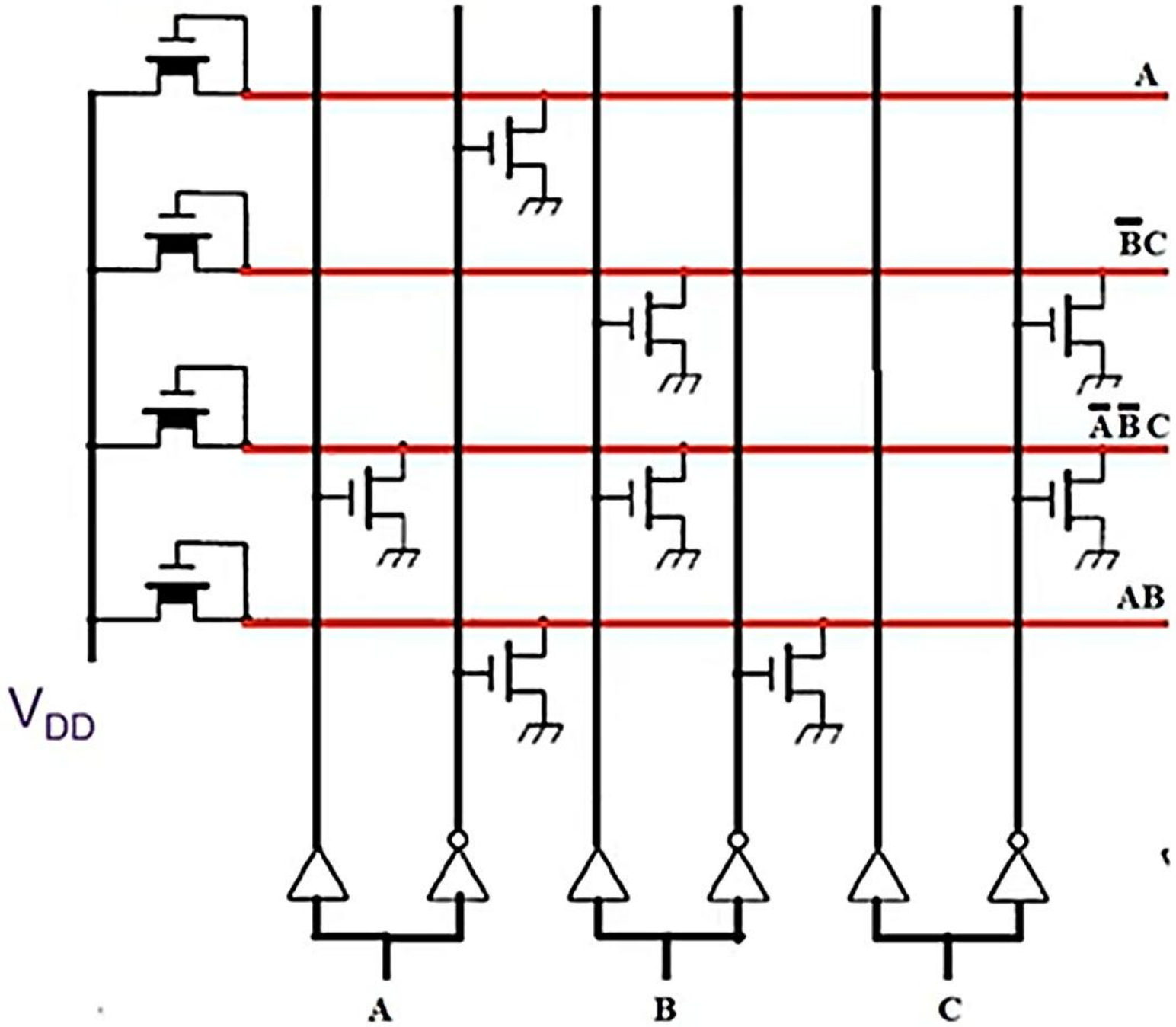


Stage 1: AND array

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

$$Y = AB$$

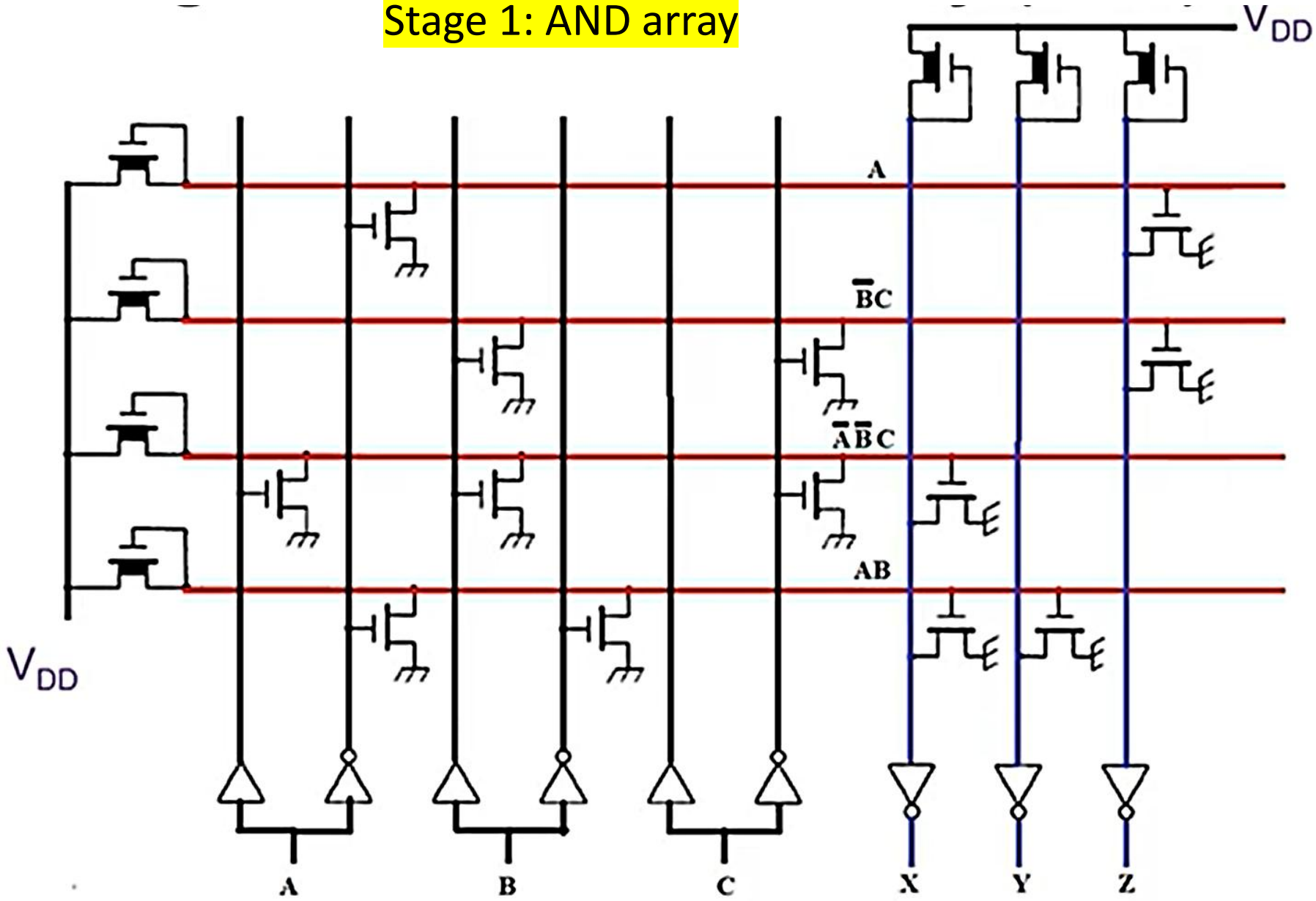
$$Z = A + \bar{B}C$$



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

$$Y = AB$$

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

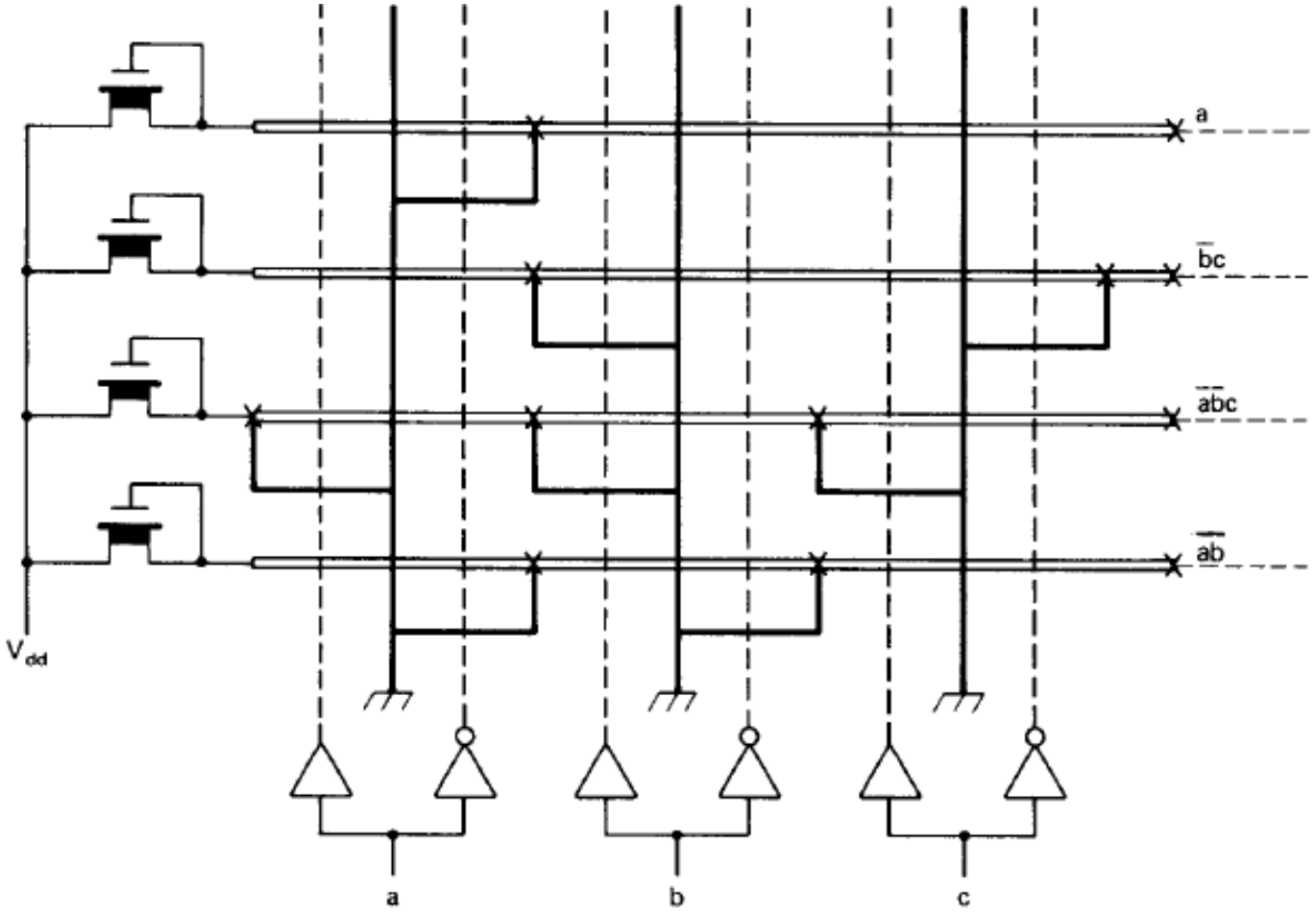


STICK DIAGRAM OF PLA

=

Stage 1: AND array

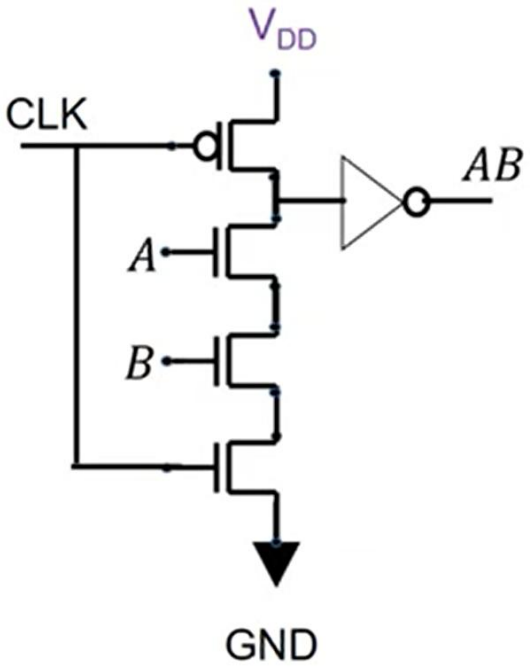
Stage 2: OR array

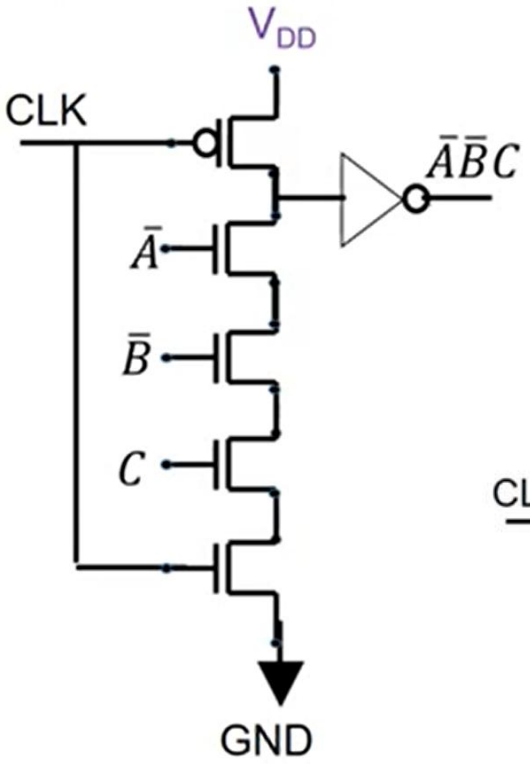


Do it for stage 2 also

2. CMOS PLA

=

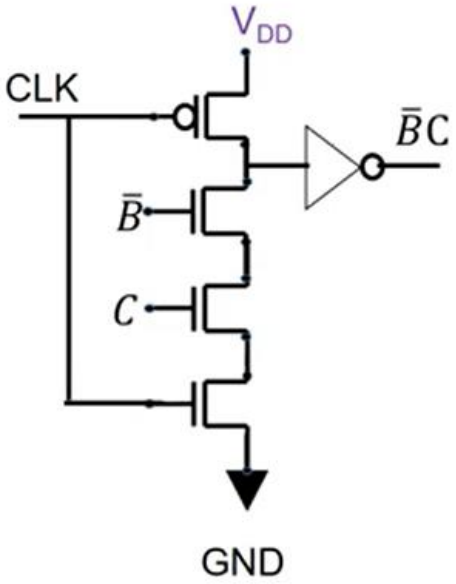


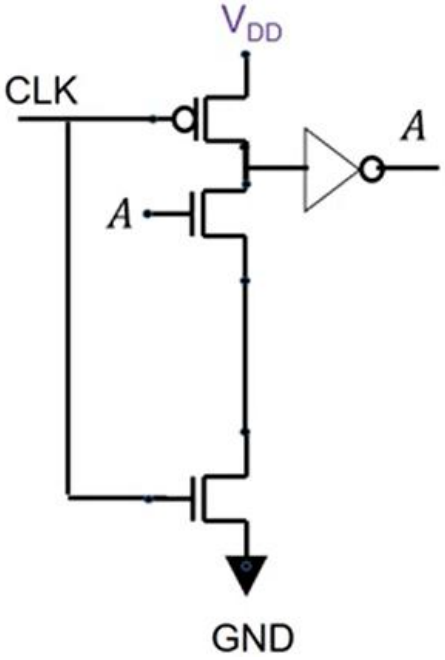


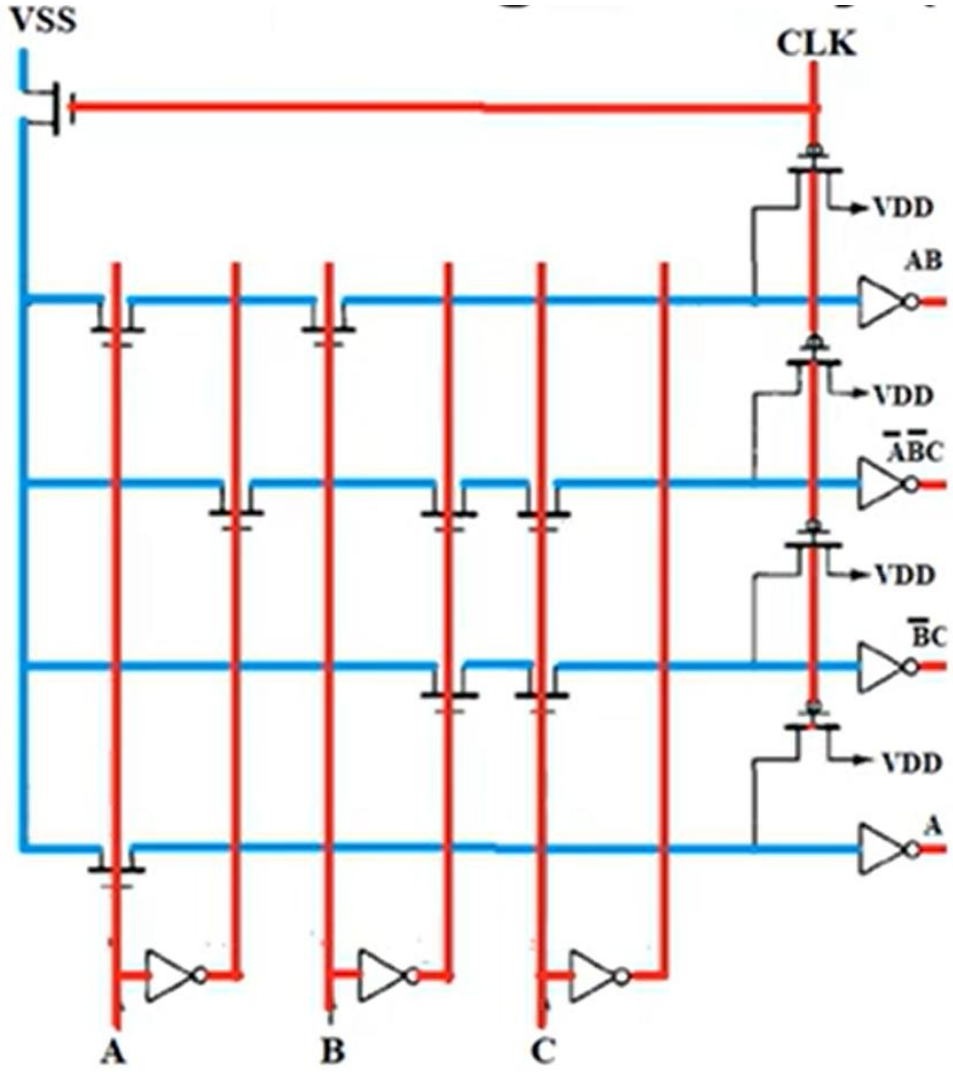
$$X = AB + \bar{A}\bar{B}C$$

$$Y = AB$$

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





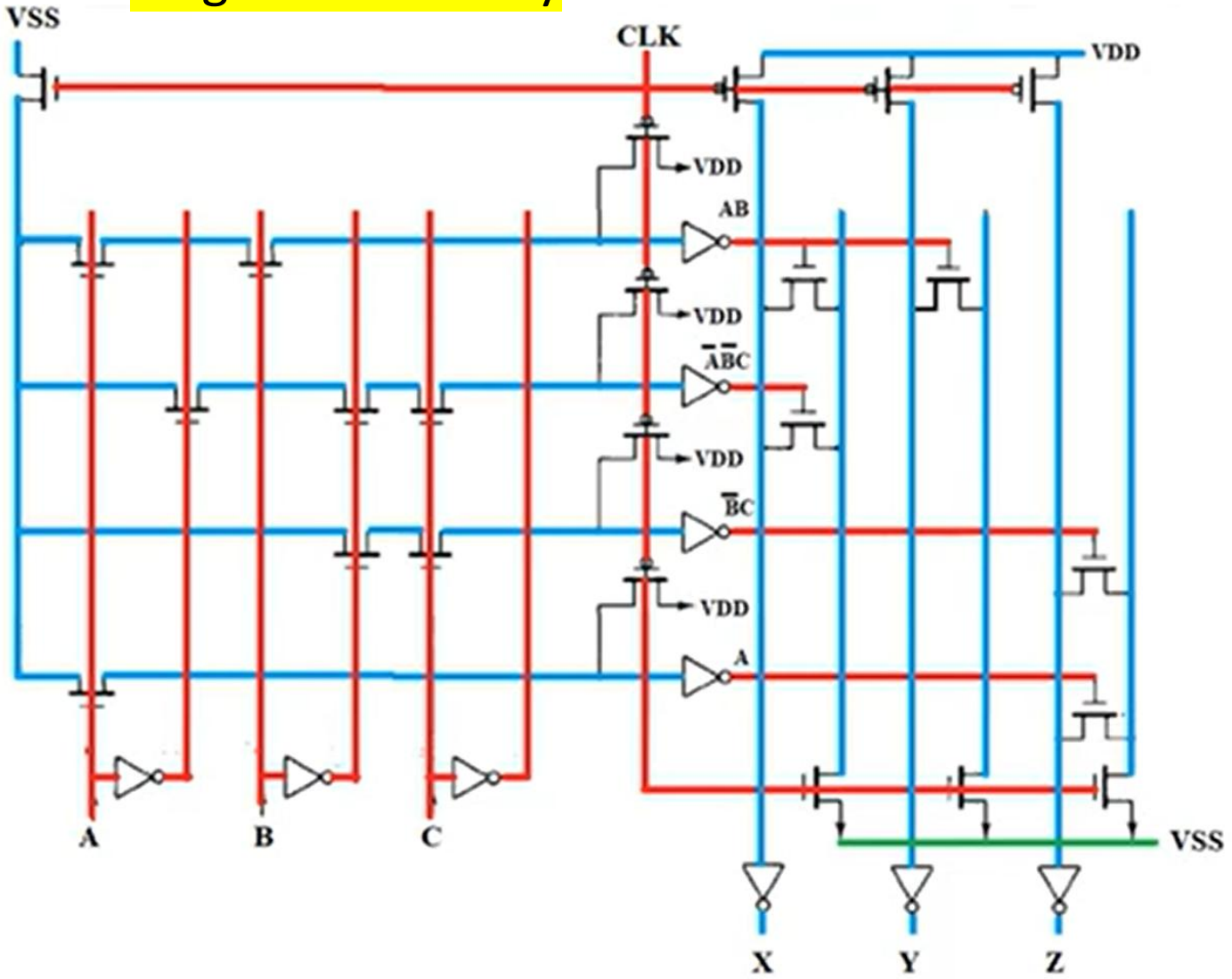


Stage 1: AND array

Stage 1: AND array

Stage 2: OR array

=



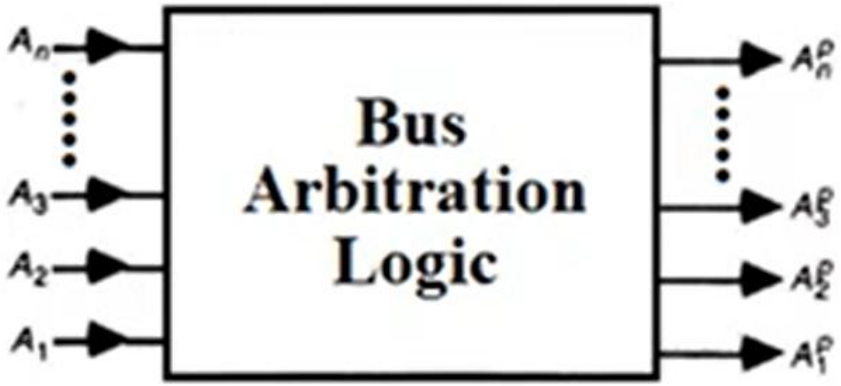
$$X = AB + \bar{A}\bar{B}C$$

$$Y = AB$$

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

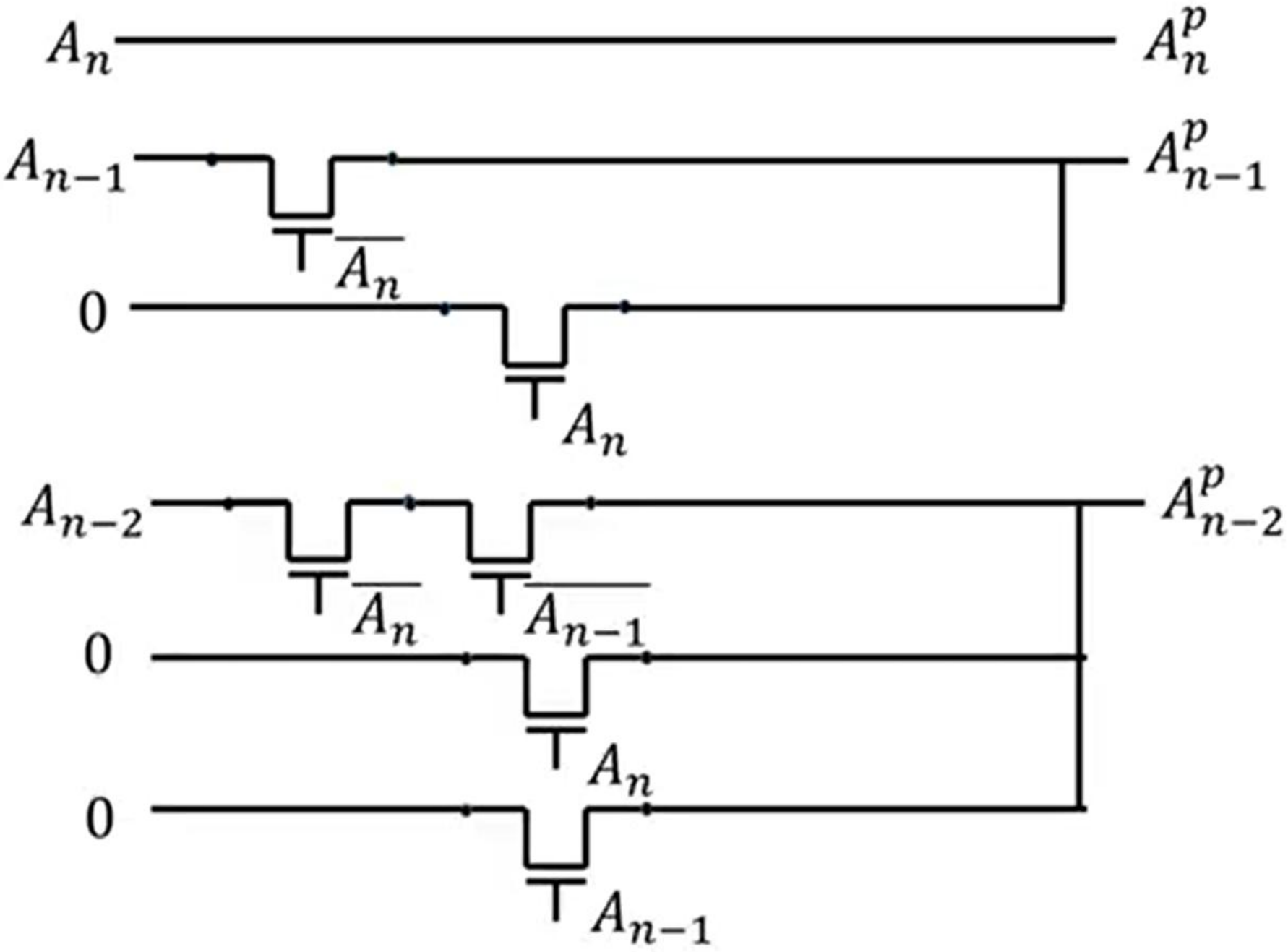
Bus Arbitration logic

Highest priority

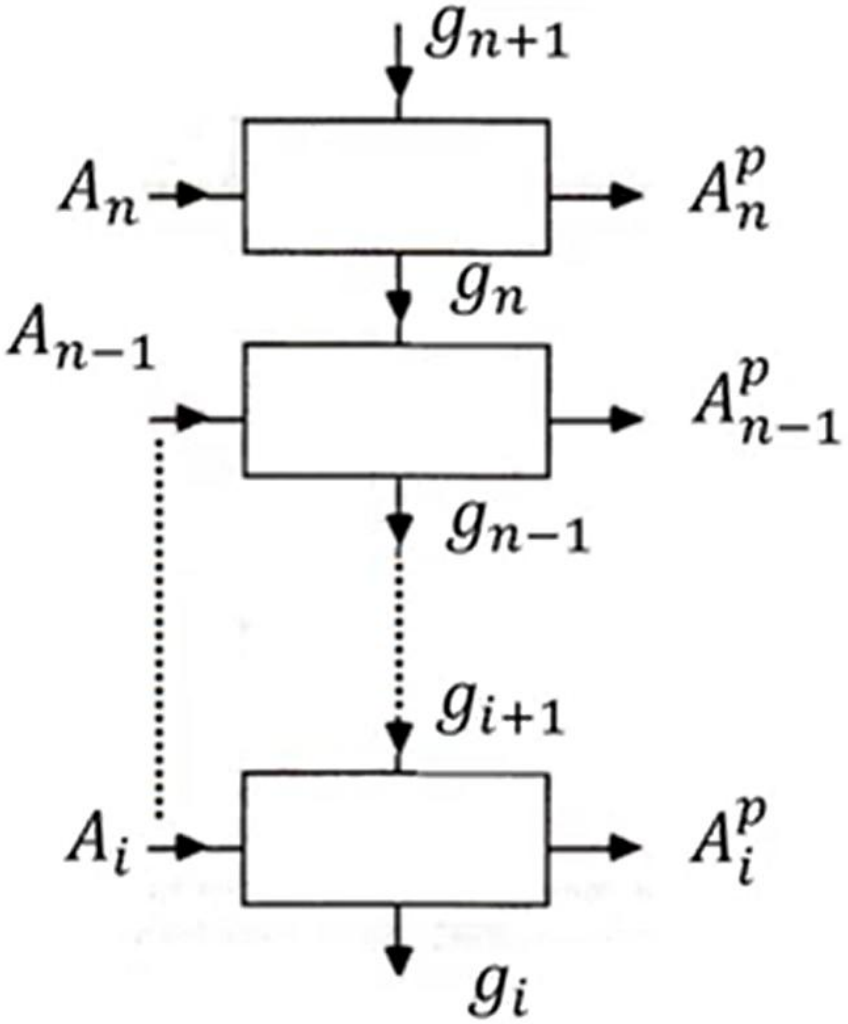


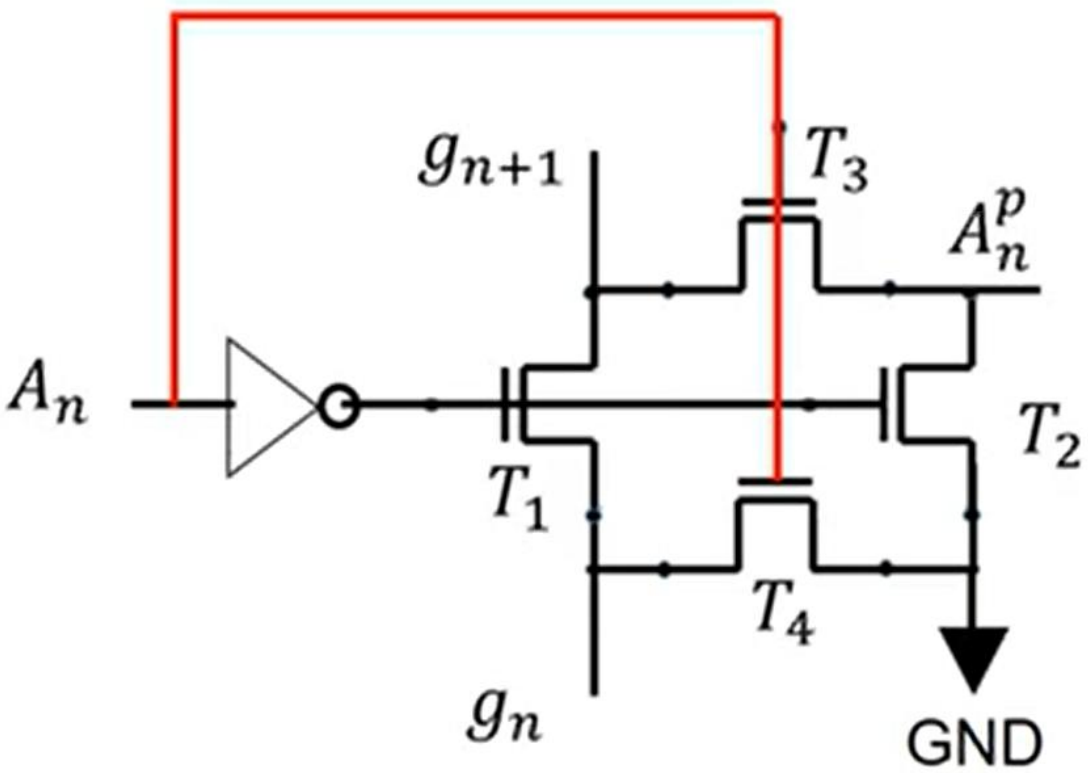
Lowest priority

A_n	•••••••	A_3	A_2	A_1	AP_n	•••••••	AP_3	AP_2	AP_1
0		0	0	0	0		0	0	0
0		0	0	1	0		0	0	1
0		0	1	X	0		0	1	0
0		1	X	X	0		1	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1		X	X	X	1		0	0	0


A_n
A_n^p
A_{n-1}
A_{n-1}^p
$\overline{A_n}$
0
A_n
A_{n-2}
A_{n-2}^p
$\overline{A_n}$
$\overline{A_{n-1}}$
0
A_n
0
A_{n-1}

=

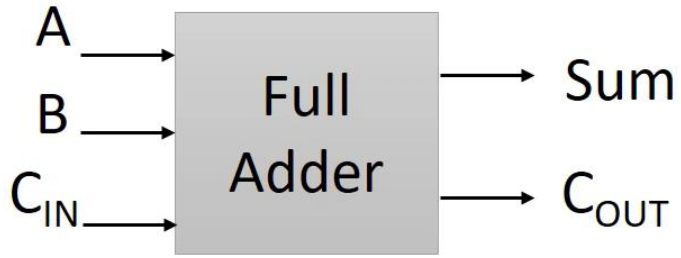




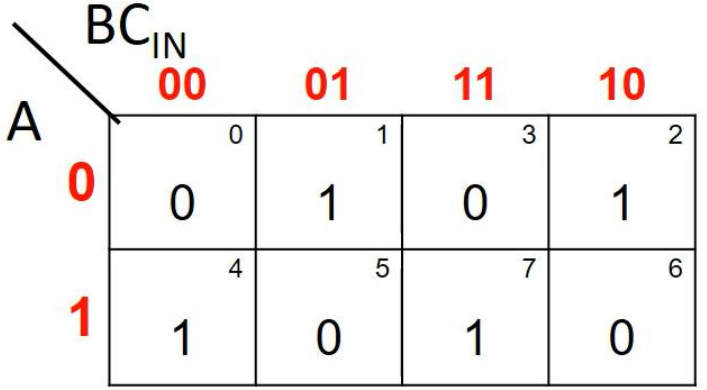
$$A_n^p = A_n g_{n+1}$$

$$g_n = g_{n+1}\overline{A_n}$$

Arithmetic Logic Unit (ALU)

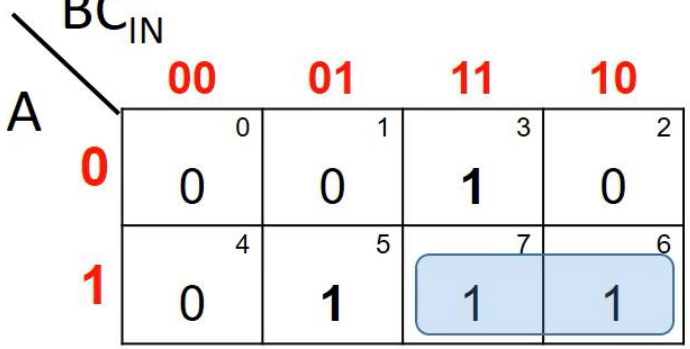


A	B	C_{IN}	Sum	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Sum= (A XOR B) XOR C_{IN}

When Cin = 0,
Then Sum= A xor B
When Cin = 1,
Then Sum= A xnor B



C_{OUT} = AB OR [(A'B OR AB') AND C_{IN}]

When Cin = 0,
Then Cout = A and B
When Cin = 1,
Then Cout = A or B

=

Sum= (A XOR B) XOR C_{IN}

If C_{IN}=0

Sum= (A XOR B) XOR 0

Sum= (A XOR B) → XOR Function

If C_{IN}=1

Sum= (A XOR B) XOR 1

Sum= (A XNOR B) → XNOR Function

C_{OUT} = AB OR [(A'B OR AB') AND C_{IN}]

If C_{IN}=0

C_{OUT} = AB OR [(A'B OR AB') AND 0]

C_{OUT} = AB → AND Function

If C_{IN}=1

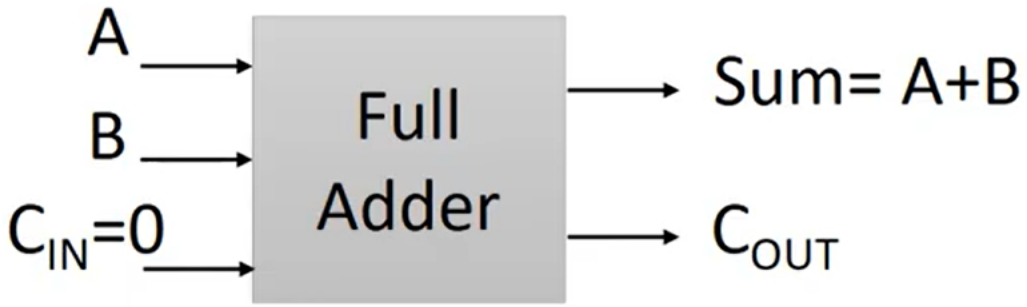
C_{OUT} = AB OR [(A'B OR AB') AND 1]

C_{OUT} = AB OR A'B OR AB'

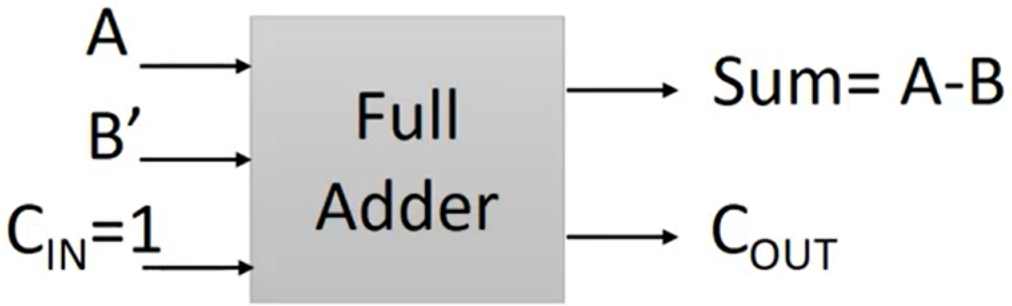
C_{OUT} = A OR B → OR Function

=

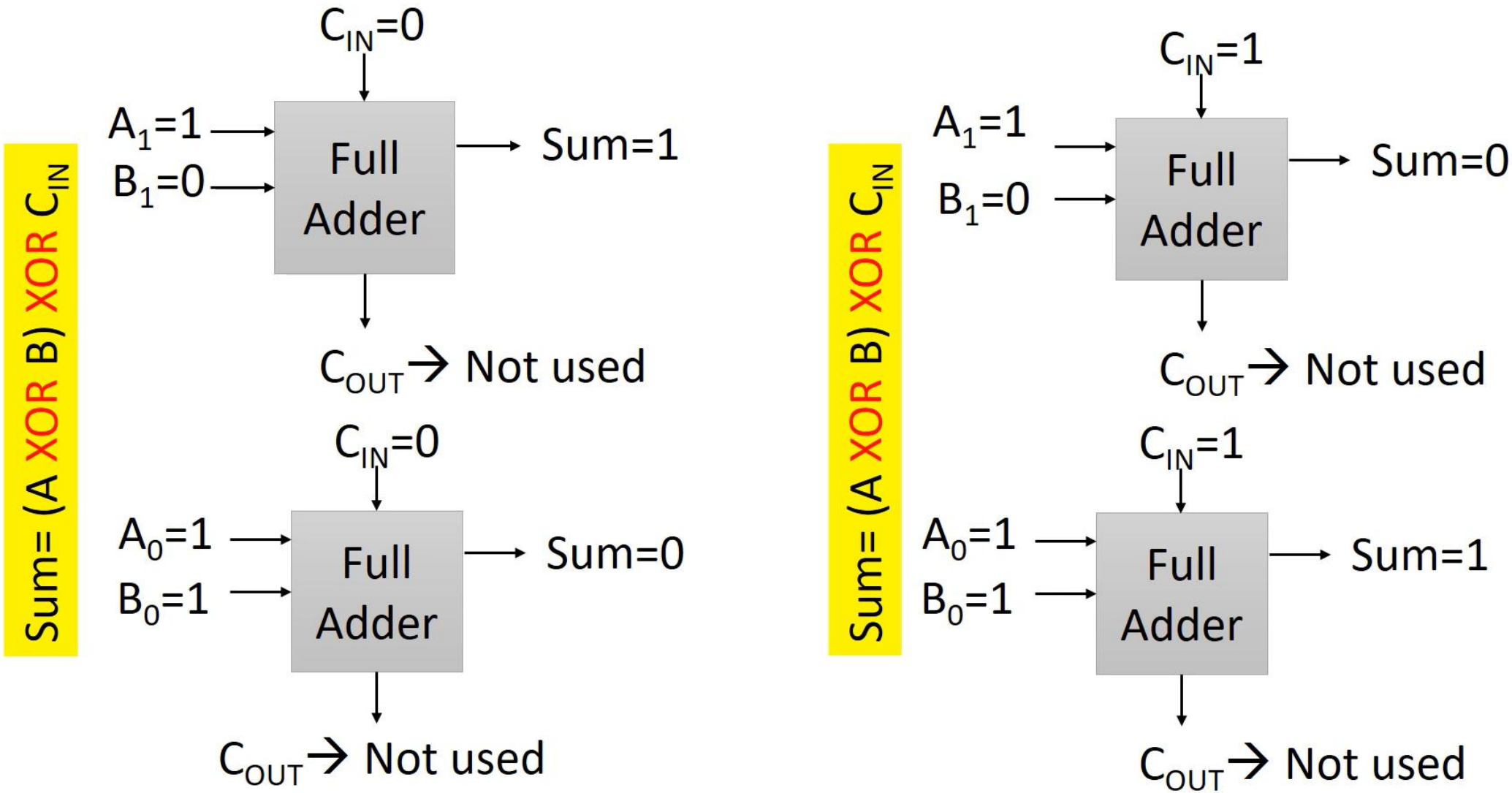
Addition



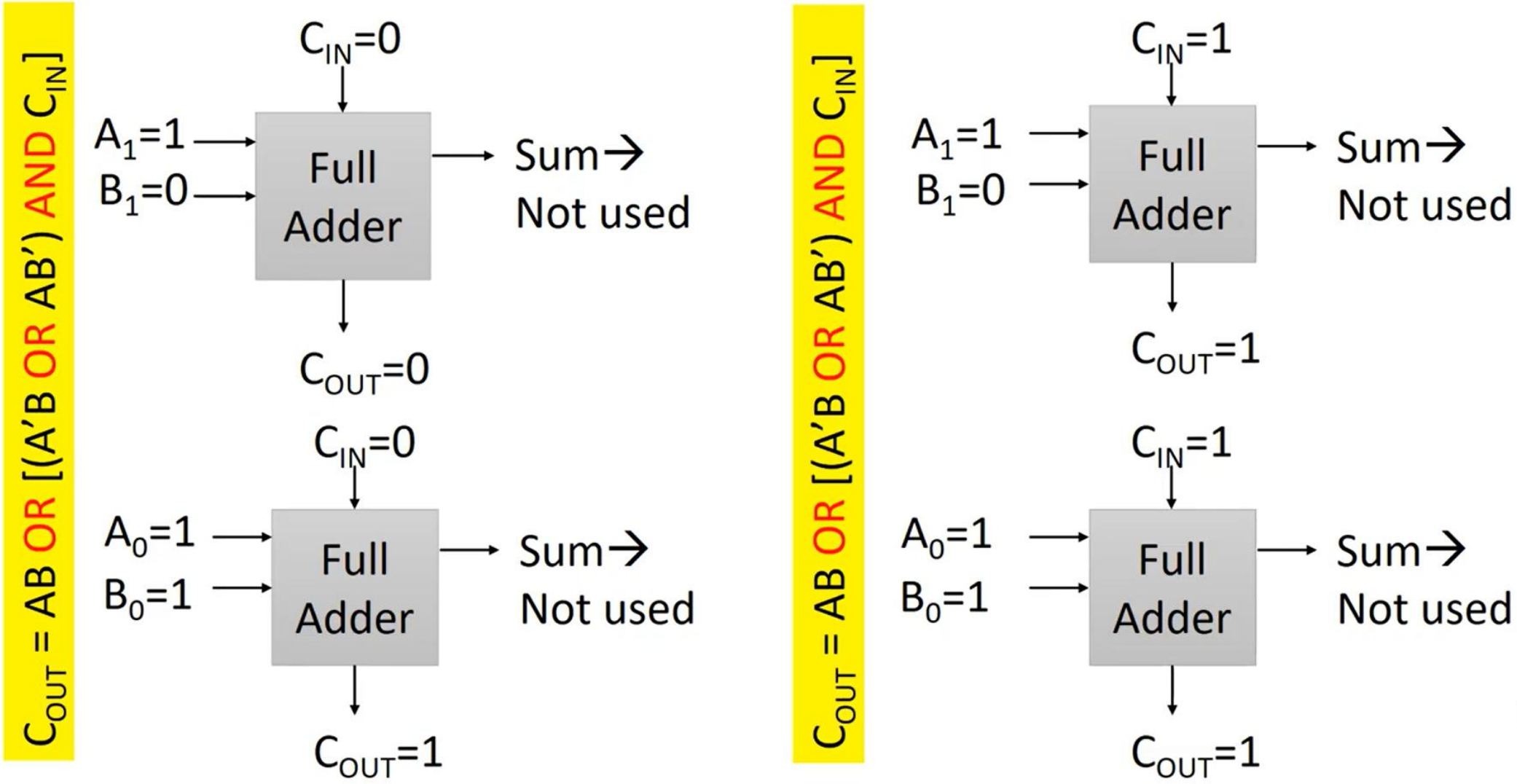
Subtraction



A-B= A+(B'+1)

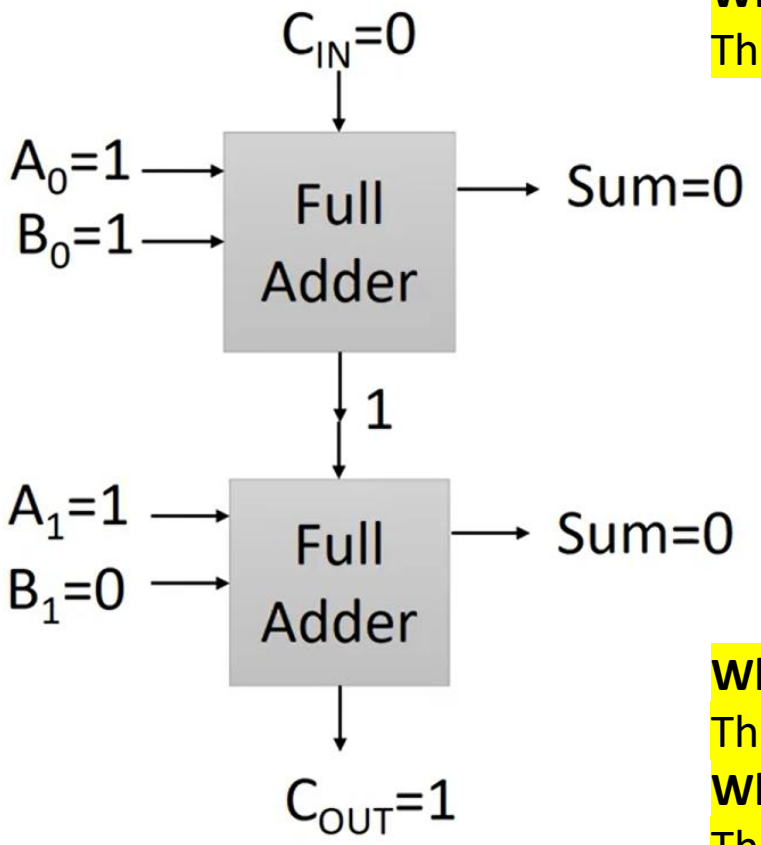

Sum= (A XOR B) XOR C_{IN}
C_{IN}=0
A_1=1
B_1=0
Full Adder
Sum=1
C_{OUT}→ Not used
C_{IN}=0
A_0=1
B_0=1
Full Adder
Sum=0
C_{OUT}→ Not used
Sum= (A XOR B) XOR C_{IN}
C_{IN}=1
A_1=1
B_1=0
Full Adder
Sum=0
C_{OUT}→ Not used
C_{IN}=1
A_0=1
B_0=1
Full Adder
Sum=1
C_{OUT}→ Not used

=


C_{OUT} = AB OR [(A'B OR AB') AND C_{IN}]
C_{IN}=0
A_1=1
B_1=0
Full Adder
Sum→
Not used
C_{OUT}=0
C_{IN}=0
A_0=1
B_0=1
Full Adder
Sum→
Not used
C_{OUT}=1
C_{OUT} = AB OR [(A'B OR AB') AND C_{IN}]
C_{IN}=1
A_1=1
B_1=0
Full Adder
Sum→
Not used
C_{OUT}=1
C_{IN}=1
A_0=1
B_0=1
Full Adder
Sum→
Not used
C_{OUT}=1

=

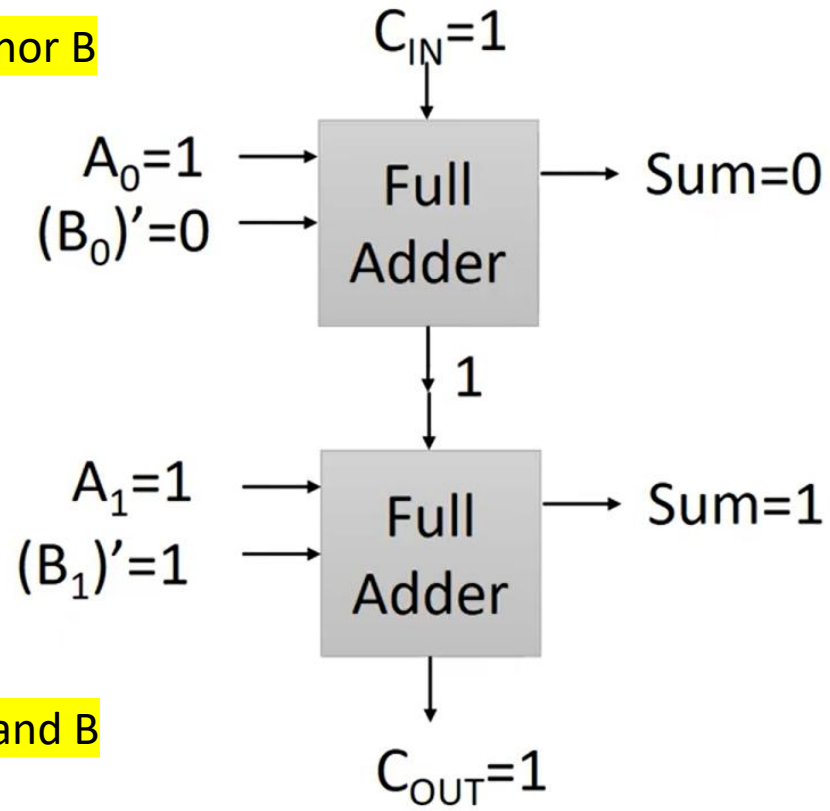
Addition



When Cin = 0,
Then Sum= A xor B
When Cin = 1,
Then Sum= A xnor B

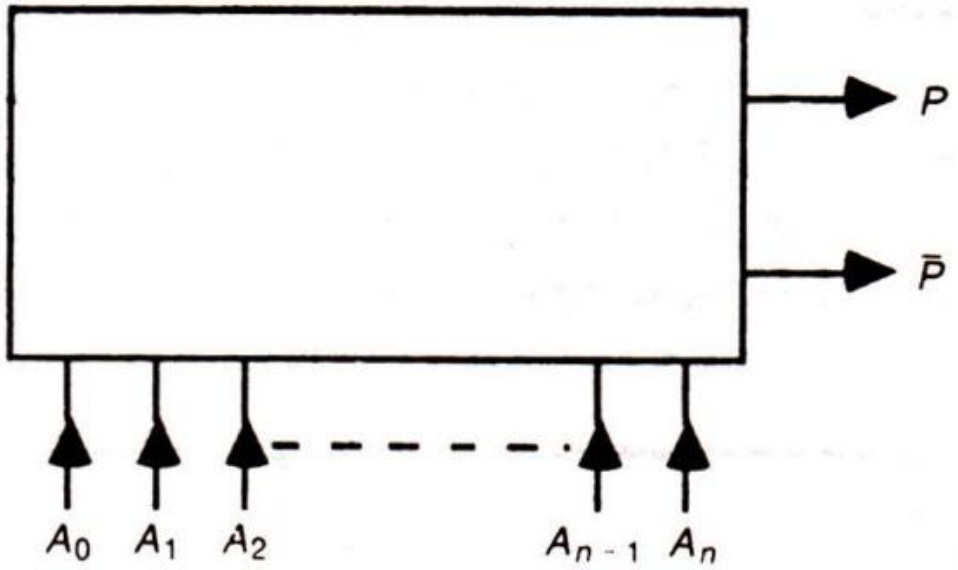
When Cin = 0,
Then Cout = A and B
When Cin = 1,
Then Cout = A or B

Subtraction



Parity generator

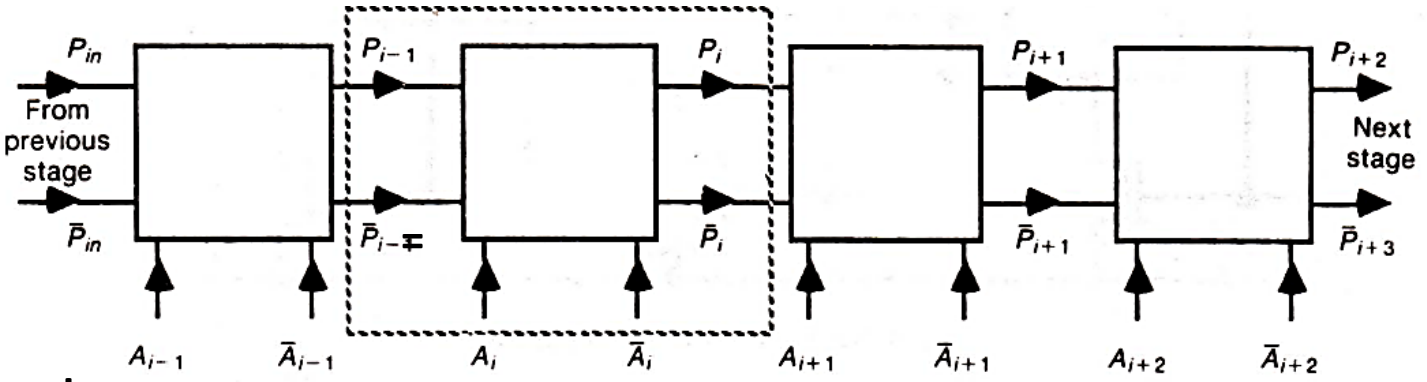
- A circuit is to be designed to indicate the parity of a binary number or word.
- The requirement is indicated in Figure 6.15 for an (n + 1)-bit input.



Note: $P = \begin{cases} 1 & \text{Even number of 1s at input} \\ 0 & \text{Odd number of 1s at input} \end{cases}$

FIGURE 6.15 Parity generator basic block diagram.

- Since the number of bits is undefined, we must find a general solution on a cascadable bit-wise basis so that “n” can have any value.
- A suitably regular structure is set out in Figure 6.16.
- From this, we may recognize_ a standard or basic one-bit cell from which an n-bit parity generator may be formed. Such a cell is shown in Figure 6.17.



Note: Parity requirements are set at the left-most cell where $P_{in} = 1$ sets even and $P_{in} = 0$ sets odd parity.

FIGURE 6.16 Parity generator—structured design approach.

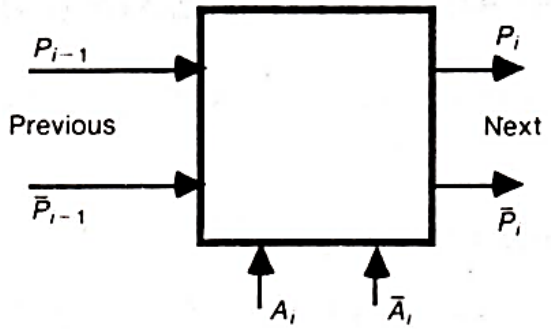
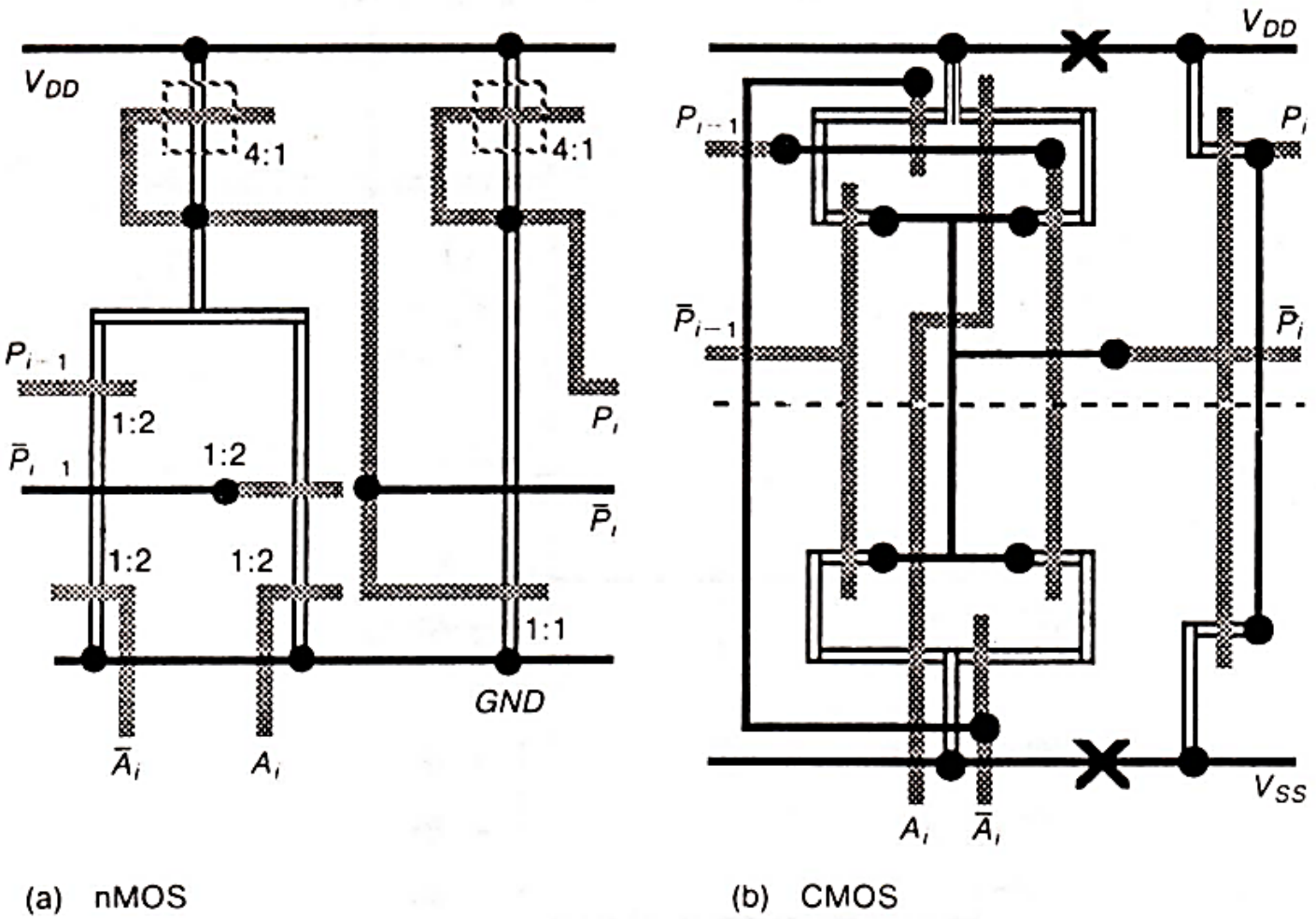


FIGURE 6.17 Parity generator—basic one-bit cell.

$A_i = 1$ parity is changed, $P_i = \bar{P}_{i-1}$
$A_i = 0$ parity is unchanged, $P_i = P_{i-1}$

$$P_i = \bar{P}_{i-1} \cdot A_i + P_{i-1} \cdot \bar{A}_i$$



(a) nMOS (b) CMOS

FIGURE 6.18 Stick diagrams (parity generator).

$$P_i = \bar{P}_{i-1} \cdot A_i + P_{i-1} \cdot \bar{A}_i$$

Shifter

=

- Any general purpose n-bit shifter should be able to shift incoming data by up to n - 1 places in a right-shift or left-shift direction.

The shifter must have:

- input from a four-line parallel data bus;
- four output lines for the shifted data;
- means of transferring input data to output lines with any shift from zero to three bits inclusive.

CROSSBAR SWITCH

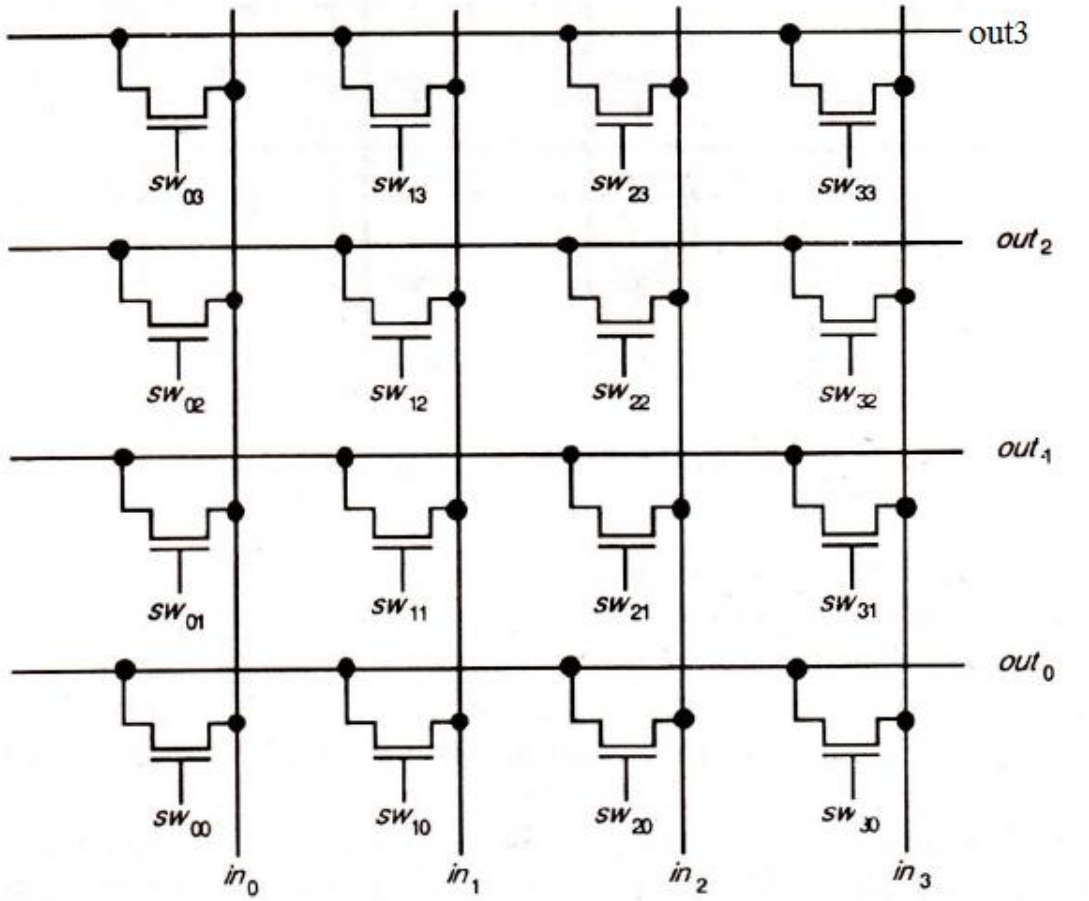
BARREL SHIFTER

Crossbar switch

=

- The attempt to design shift register using cross bar switch (like MOS pass transistor and transmission gate etc.)
- We must have the strategy in which data flow horizontally and control signals are kept vertically.
- switch and relay contact based switching networks is called -the *crossbar switch.*

MOS switch
implementation of a 4 x 4 crossbar switch



If all switches are on then all input are connected to output and short circuit is possible. We may take the combination of switches to shifts of zero, one, two and three bits.

4 x 4-bit barrel shifter

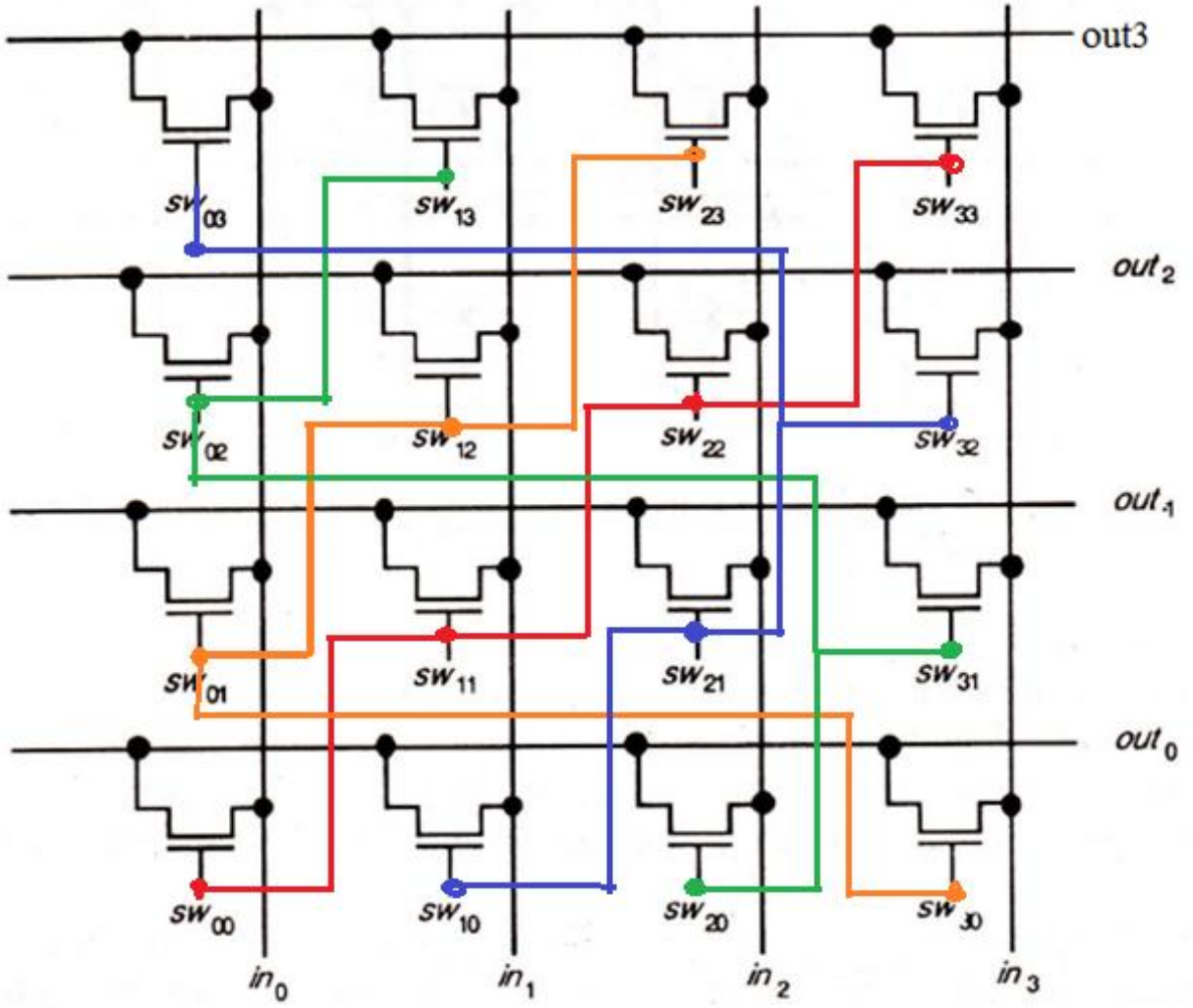
=

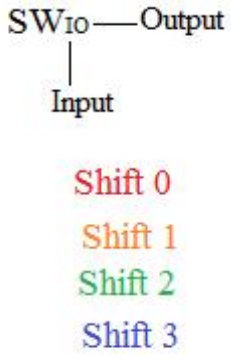
- *A barrel shifter is a digital circuit that can shift a data word by a specified number of bits in each clock cycle.*
- The resulting arrangement is known as a *barrel shifter* as shown below (next slide).

Out 0	Out 1	Out 2	Out 3	
In 0	In1	In2	In3	Shift 0 (eg. 1001)
In3	In0	In1	In2	Shift1 (eg. 1100)
In2	In3	In0	In1	Shift2 (eg. (0110)
In1	In2	In3	In0	Shift3 (eg. (0011)

4 x 4-bit barrel shifter

=





No of rows = word length of data
No of column= maximum shift in data