9.5 ROM (Read-Only Memory)

* Stores binary data
* Data can be read out whenever desired
* Data can not be changed under normal operating conditions

Uses of ROM

Storing Programs:

Computer booting (bootstraping) firmware

Firmware for selfcontained devices (graphic cards, hard disks, DVD drives, etc.)

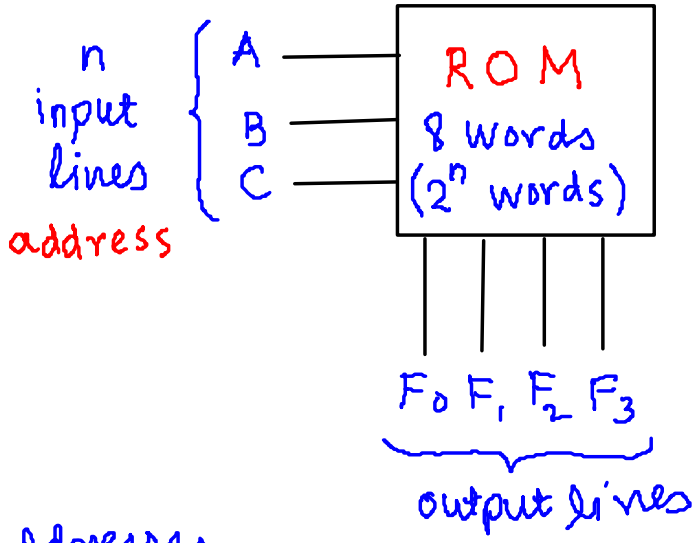
Embedded systems (industrial robots, home appliances, consumer electronics mp3 players, cameras)

Storing Data:

Store lookup tables that don't need modification (read only)

Mass storage (USB drives)

Sophomore Lab sine, square, sawtooth table for signal generator, key board etc.



n addresses

A	B	C	F_0	F_1	F_2	F_3
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	1	0	1	1	1	1
1	1	1	0	1	0	1

Stored Data

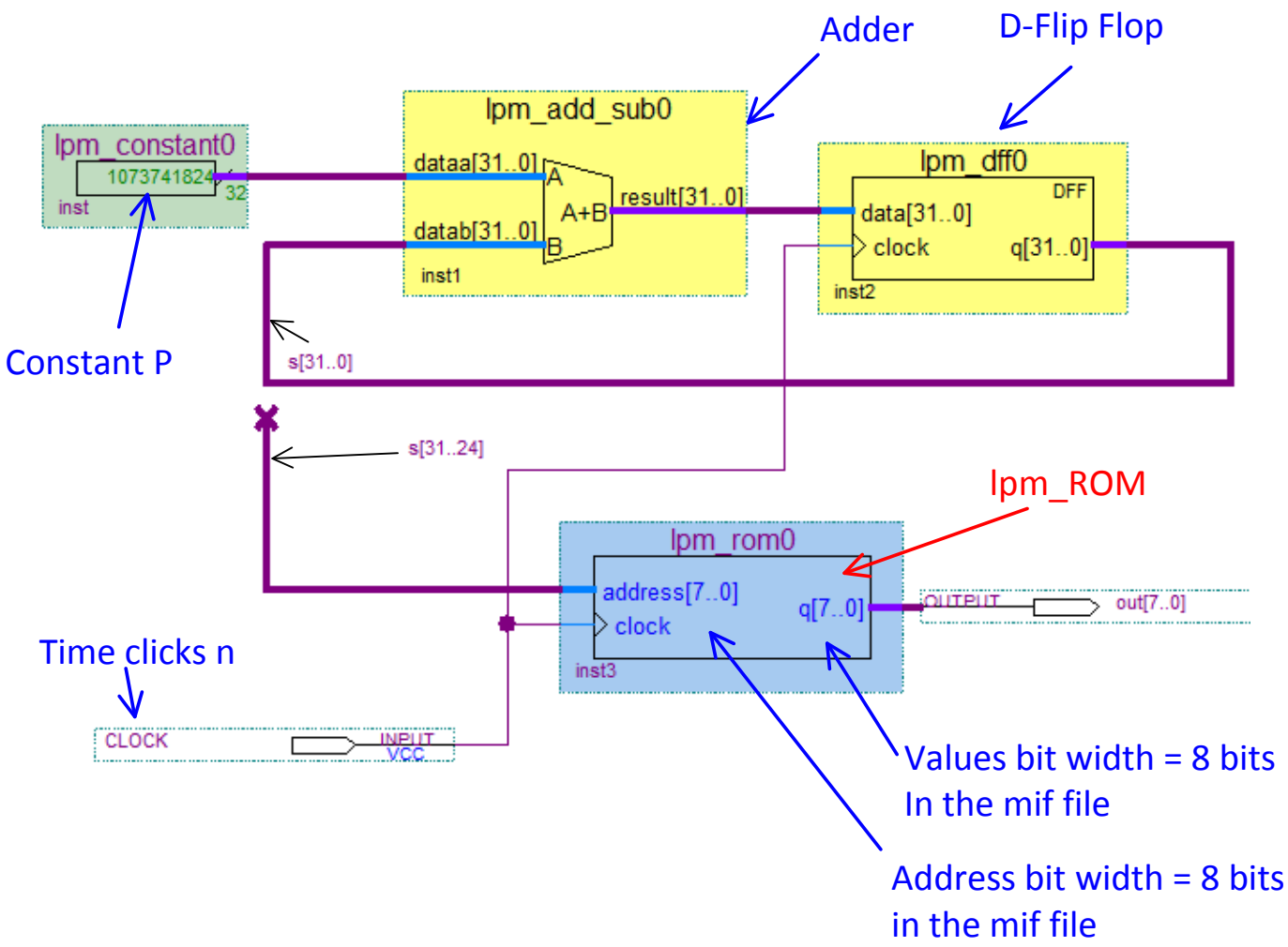
2^n words

Look up table

Word width = 4 bits (m bits)

Size of the ROM = # of rows × # of columns

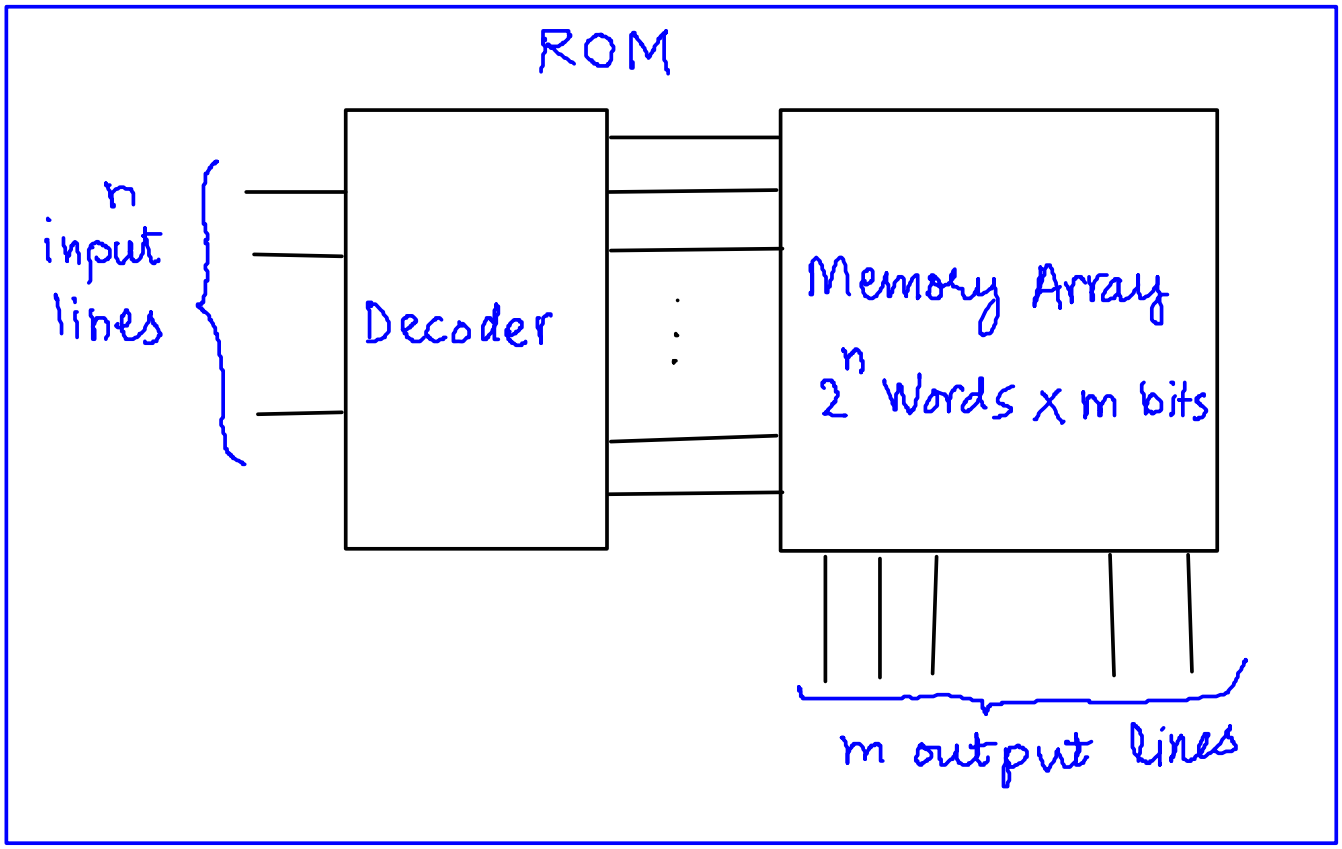
$= 2^n \times m$


Adder
D-Flip Flop
lpm_add_sub0
lpm_constant0
1073741824
32
inst
dataa[31..0]
A
A+B
result[31..0]
datab[31..0]
B
inst1
lpm_dff0
DFF
data[31..0]
clock
q[31..0]
inst2
Constant P
s[31..0]
s[31..24]
lpm_ROM
lpm_rom0
address[7..0]
clock
q[7..0]
inst3
OUTPUT
out[7..0]
Time clicks n
CLOCK
INPUT
VCC
Values bit width = 8 bits
In the mif file
Address bit width = 8 bits
in the mif file

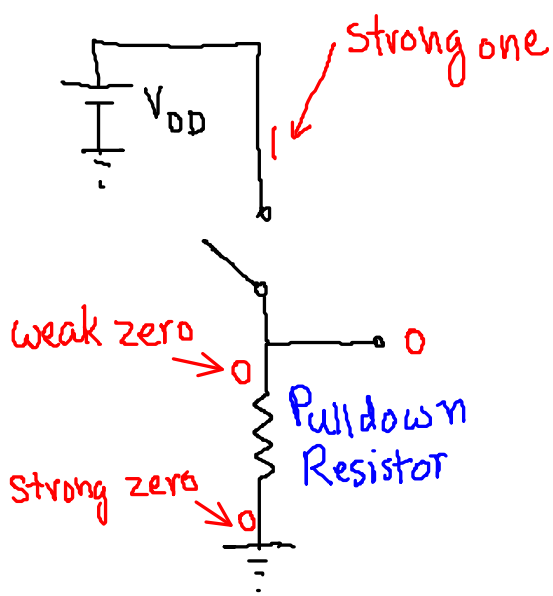
Sample mif file

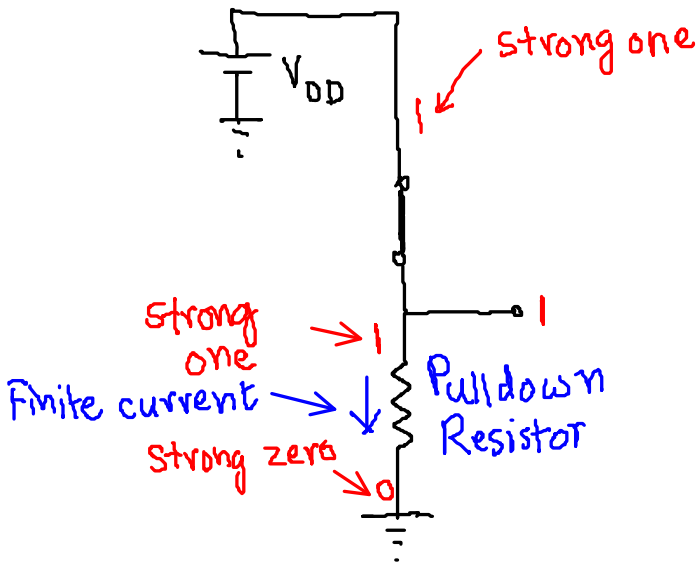
Address bit width: 8
Value bit width :16 (signed)

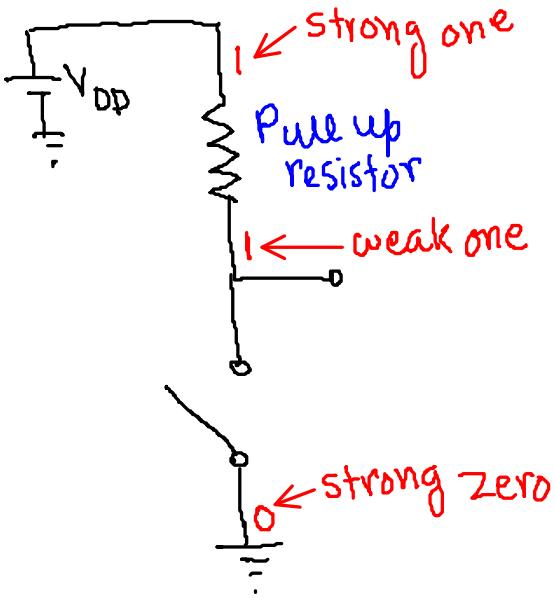
```
WIDTH=16;
DEPTH= 256;

ADDRESS_RADIX=DEC;
DATA_RADIX=DEC;

CONTENT BEGIN
  0 :    0;
  1 :  799;
  2 : 1598;
  3 : 2396;
  4 : 3192;
  5 : 3987;
  6 : 4779;
  7 : 5568;
  8 : 6353;
  9 : 7135;
 10 : 7913;

       ....

 248 : -5568;
 249 : -4779;
 250 : -3987;
 251 : -3192;
 252 : -2396;
 253 : -1598;
 254 : -799;
 255 :   -0;
END;
```

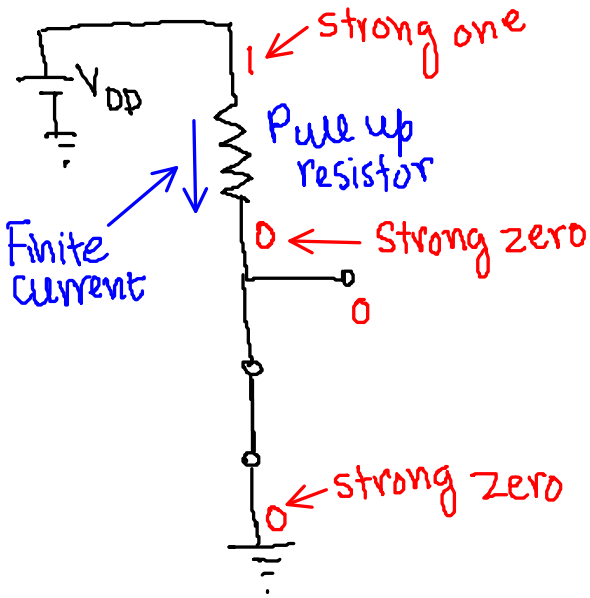

ROM
n
input
lines
Decoder
Memory Array
2^n Words x m bits
m output lines

Pull-up and pull-down resistors



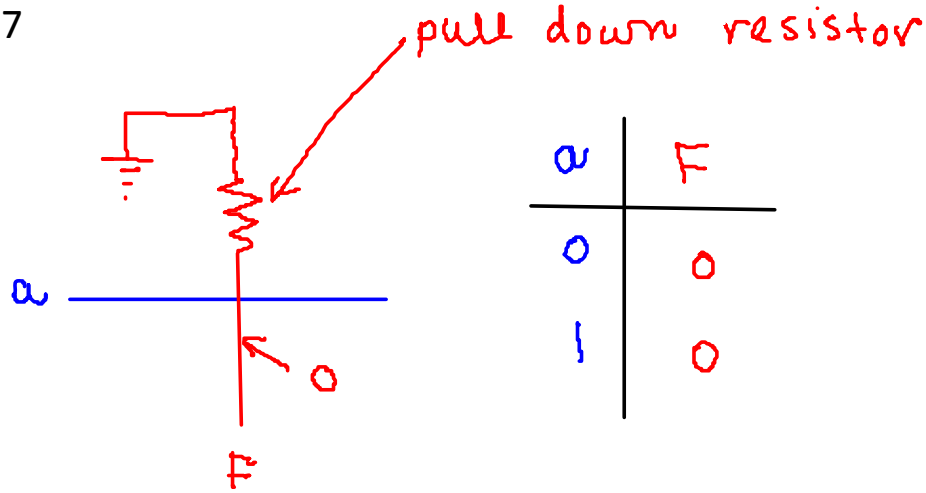




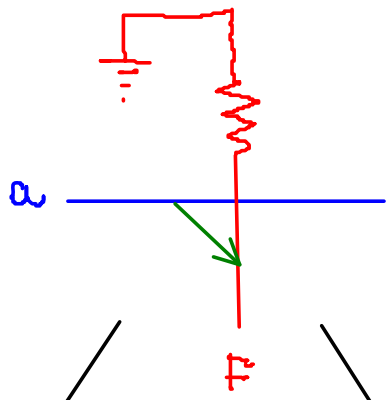


Strong 1 & strong 0 → X Strong 1 & weak 0 → 1

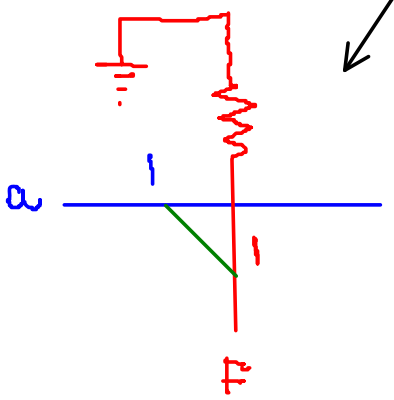
Strong 0 & weak 1 → 0 0 & 0 → 0 1 & 1 → 1



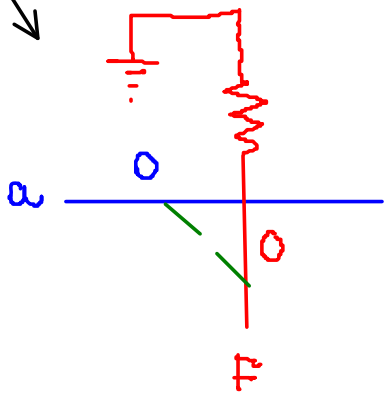
a	F
0	0
1	0



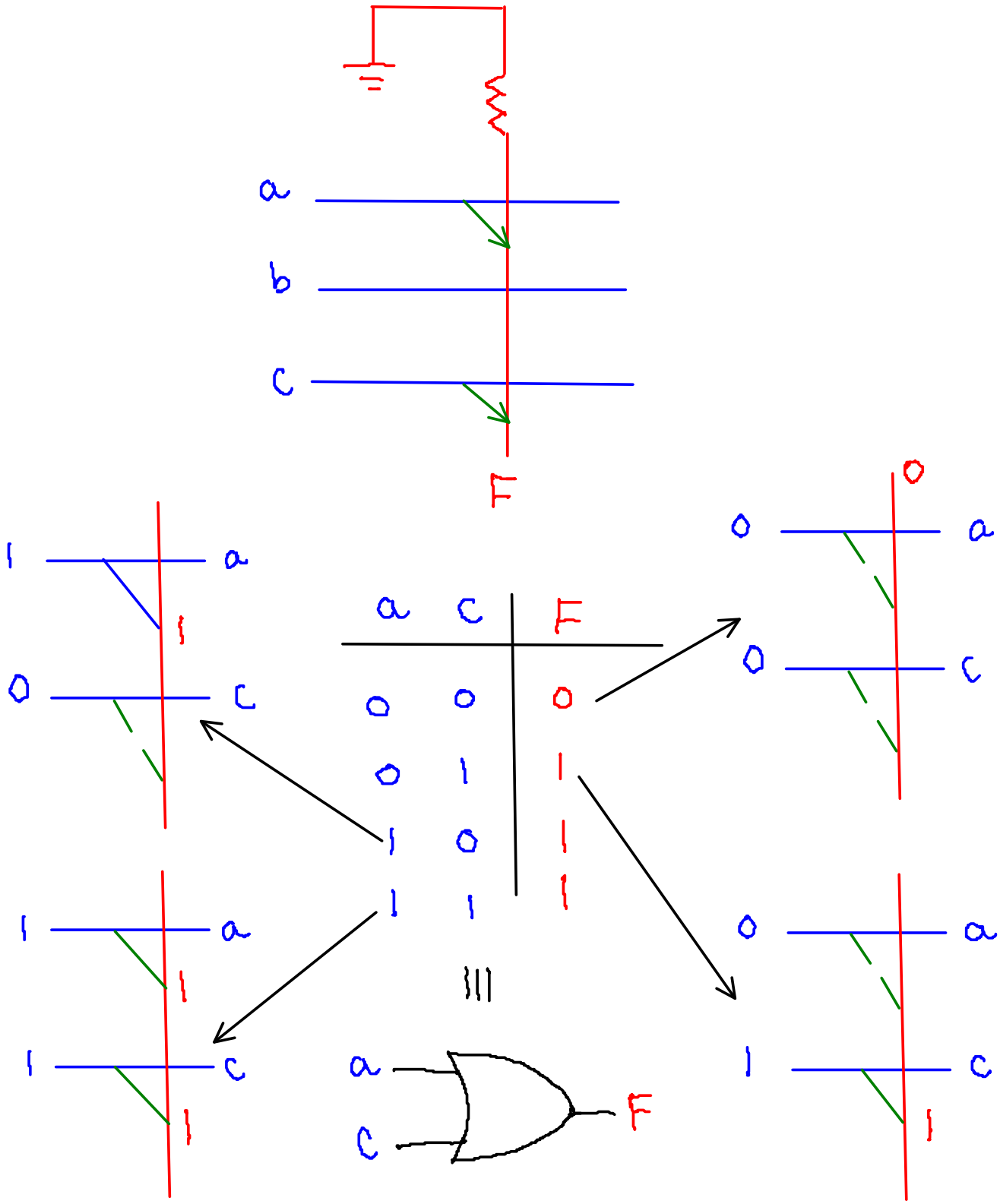
a = 1



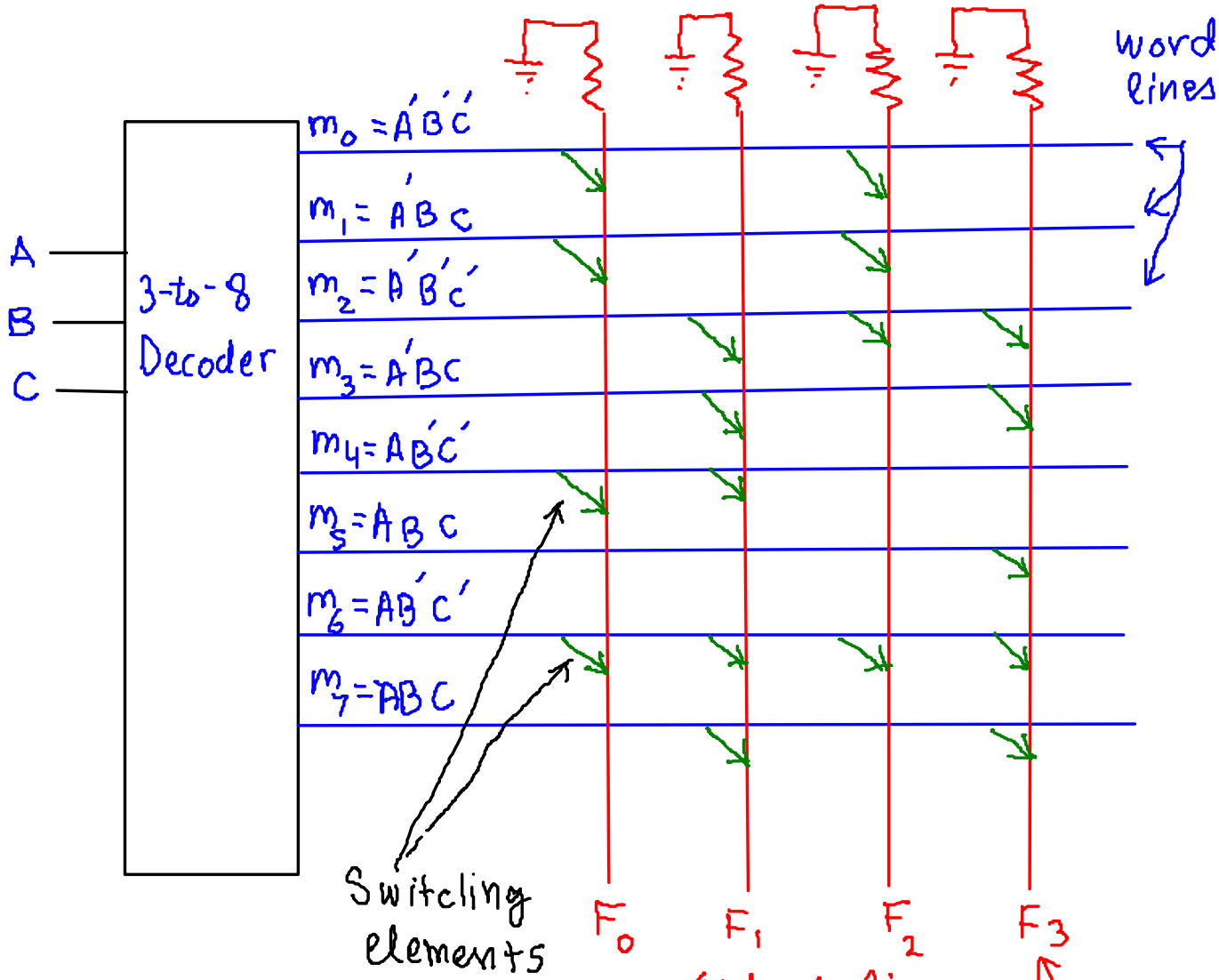
a = 0



a	F
0	0
1	1

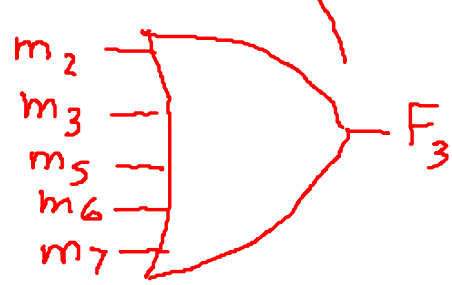

a
b
c
F
a c F
0 0 0
0 1 1
1 0 1
1 1 1
a
c
F

8 word x 4 bit ROM



Switching element connected if minterm is to be included

$F_0 = m_0 + m_1 + m_4 + m_6$

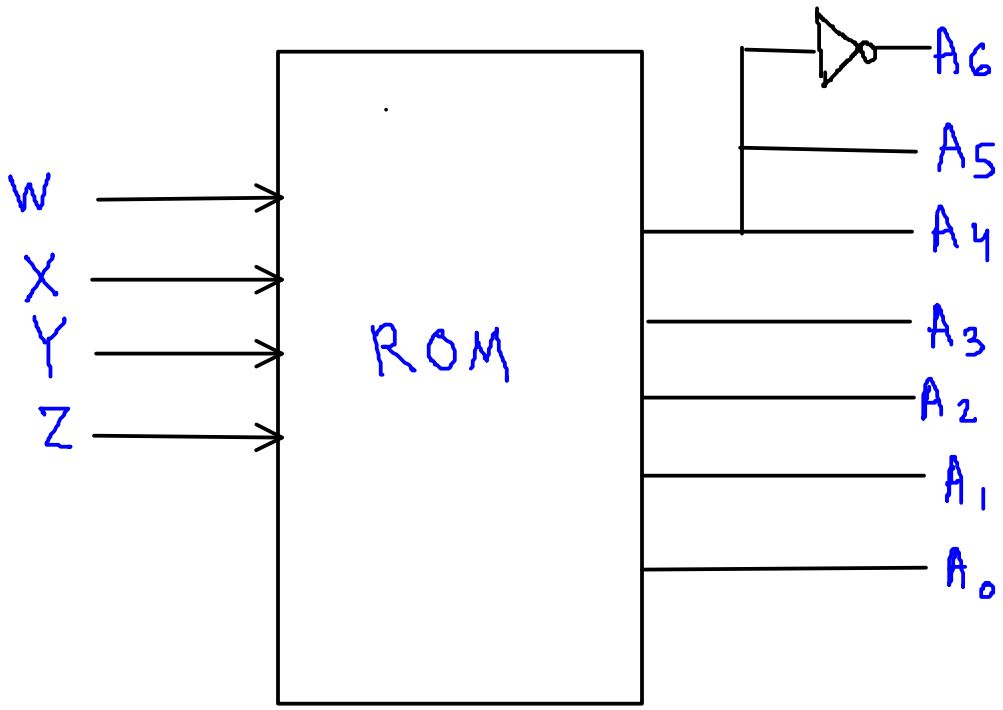


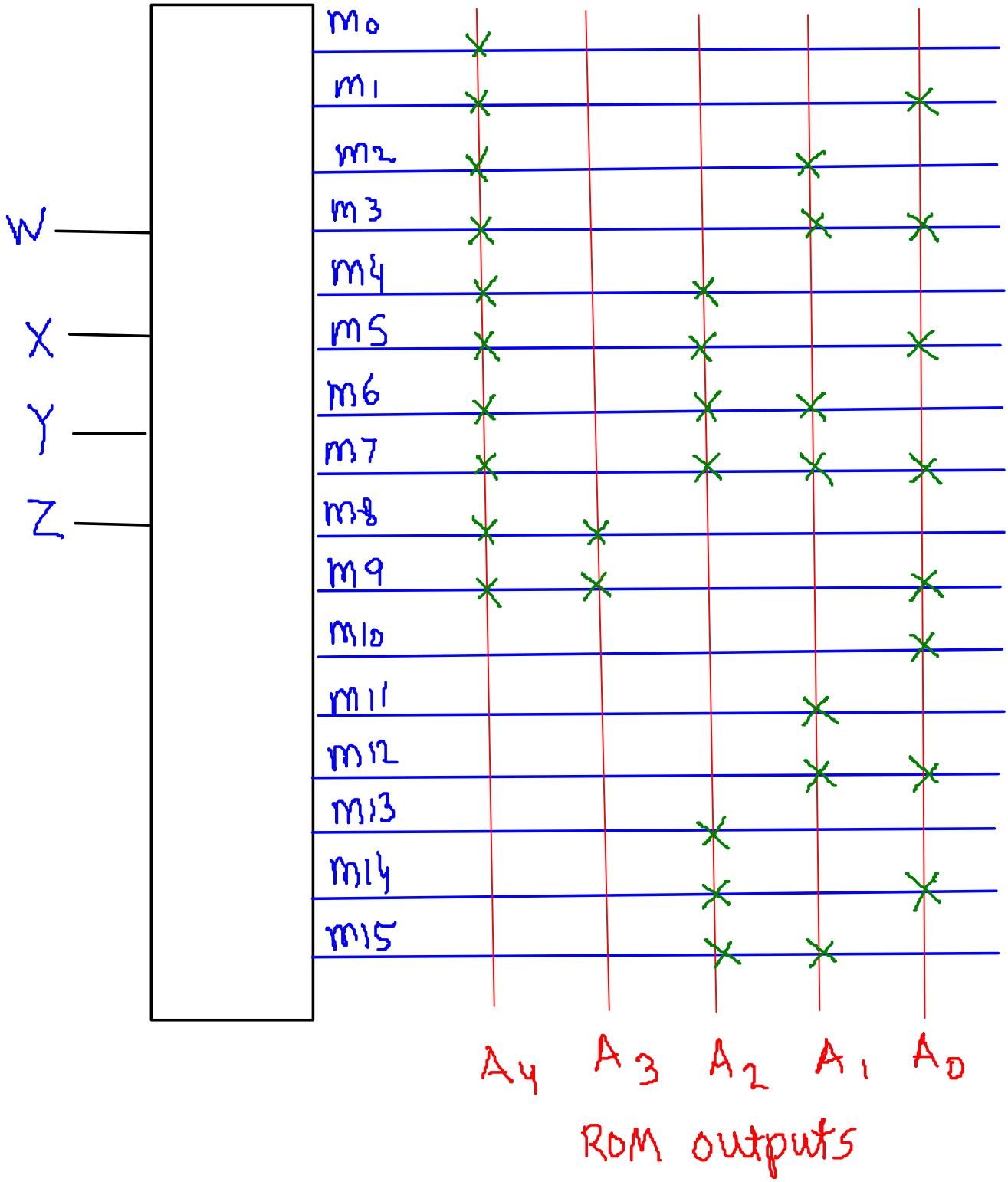
OR ARRAY

Hexadecimal-to-ASCII Code Converter

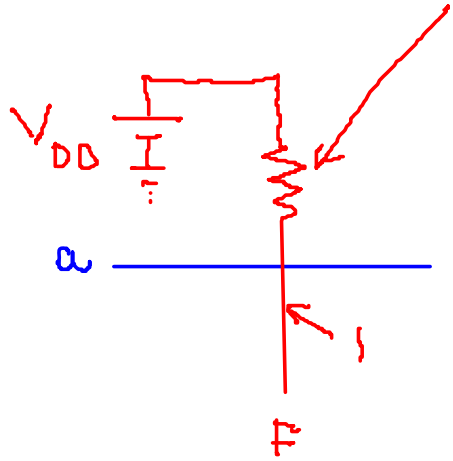
(see TABLE 1-3, page 23)

Hex	W	X	Y	Z	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	0	1	1	0	0	0	0
1	0	0	0	1	0	1	1	0	0	0	1
2	0	0	1	0	0	1	1	0	0	1	0
3	0	0	1	1	0	1	1	0	0	1	1
4	0	1	0	0	0	1	1	0	1	0	0
5	0	1	0	1	0	1	1	0	1	0	1
6	0	1	1	0	0	1	1	0	1	1	0
7	0	1	1	1	0	1	1	0	1	1	1
8	1	0	0	0	0	1	1	1	0	0	0
9	1	0	0	1	0	1	1	1	0	0	1
A	1	0	1	0	1	0	0	0	0	0	1
B	1	0	1	1	1	0	0	0	0	1	0
C	1	1	0	0	1	0	0	0	0	1	1
D	1	1	0	1	1	0	0	0	1	0	0
E	1	1	1	0	1	0	0	0	1	0	1
F	1	1	1	1	1	0	0	0	1	1	0

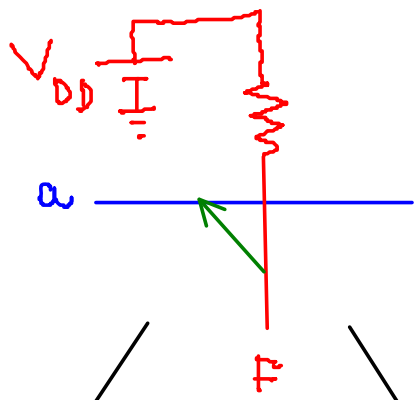

W
X
Y
Z
ROM
A6
A5
A4
A3
A2
A1
A0


W
X
Y
Z
m0
m1
m2
m3
m4
m5
m6
m7
m8
m9
m10
m11
m12
m13
m14
m15
A4
A3
A2
A1
A0
ROM outputs

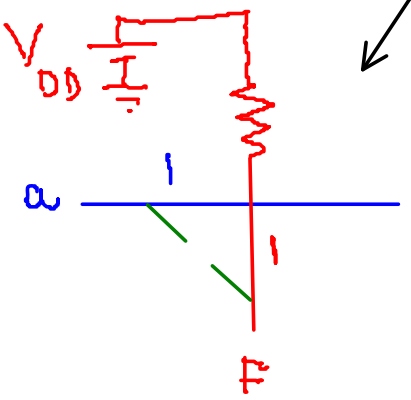
pull up resistor



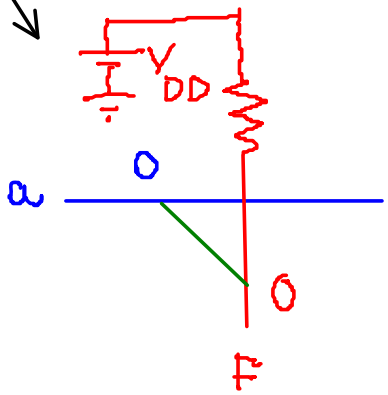
a	F
0	1
1	1



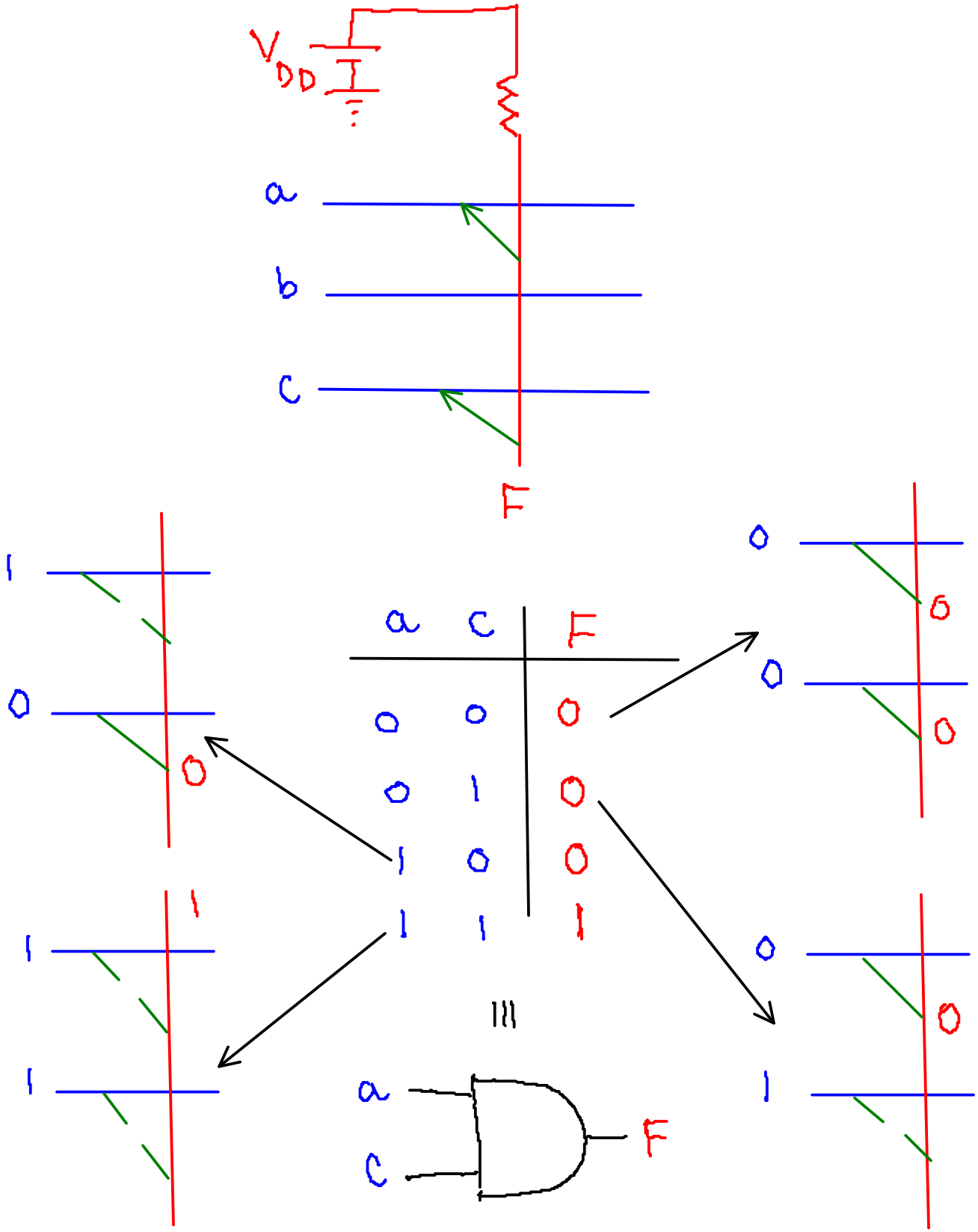
a = 1



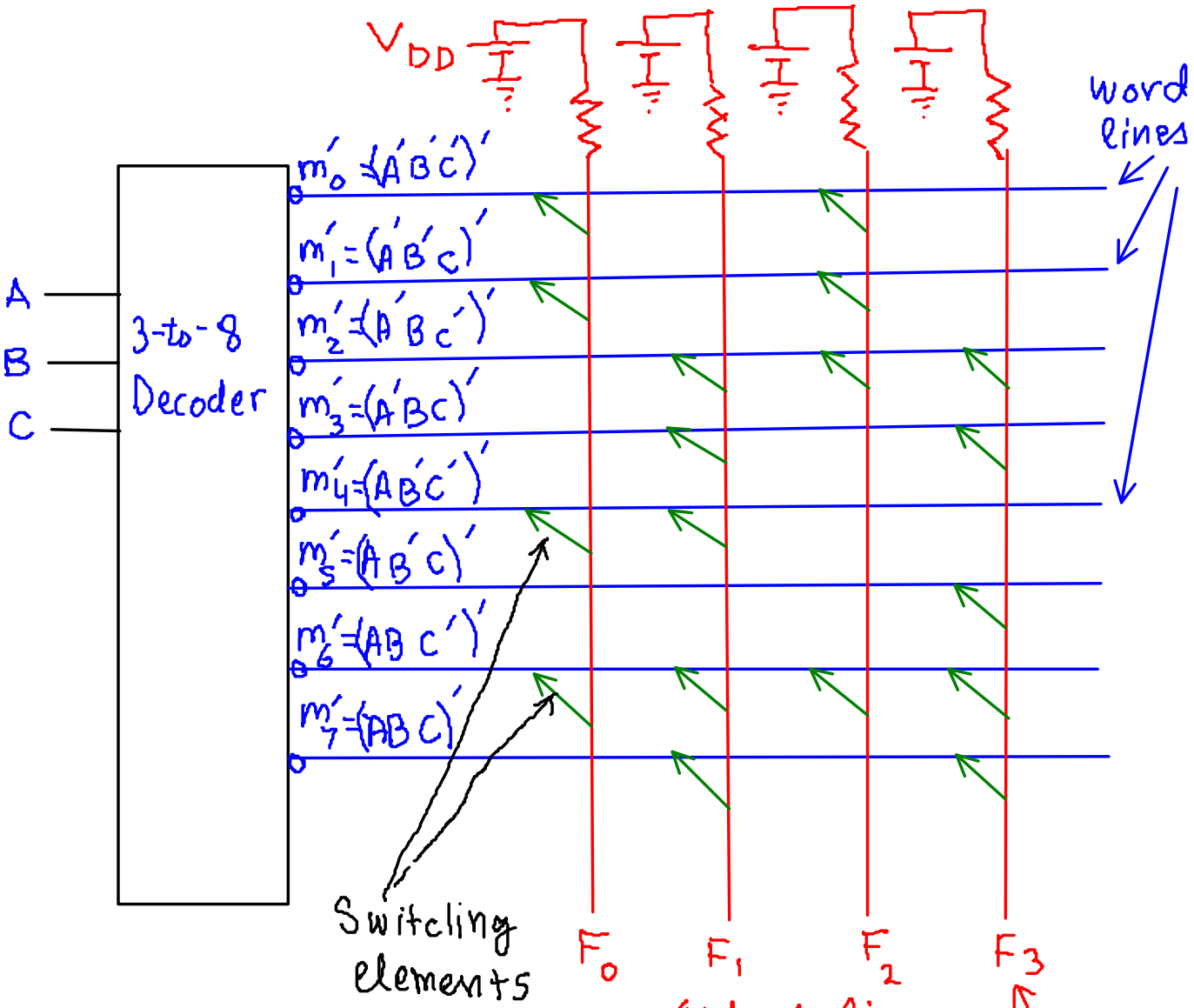
a = 0



a	F
0	0
1	1


VDD
a
b
c
F
a c F
0 0 0
0 1 0
1 0 0
1 1 1
a
c
F

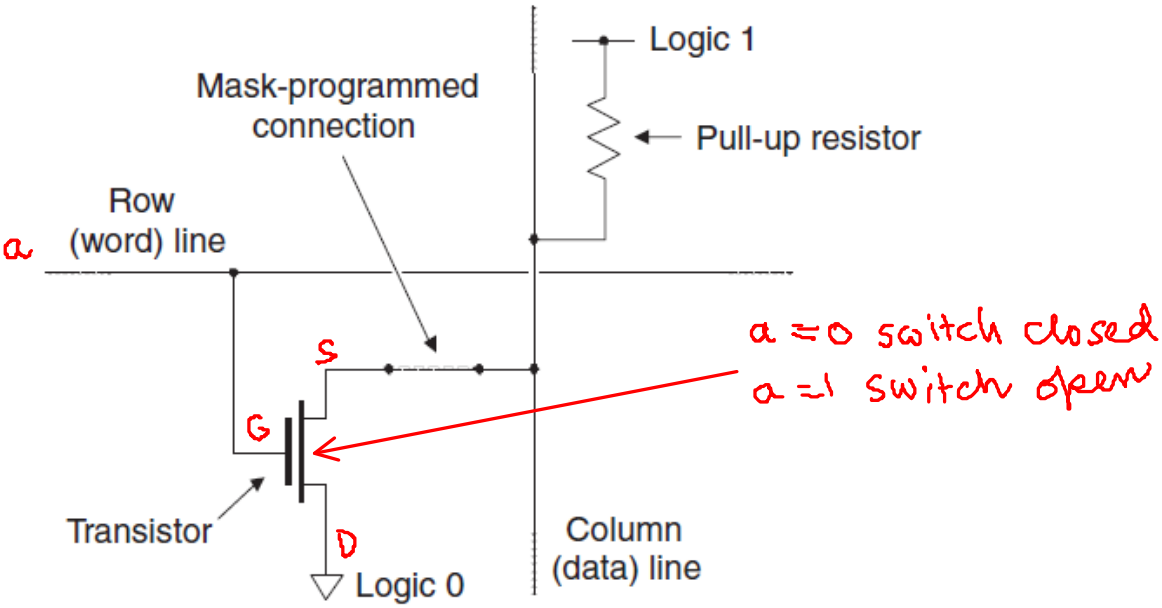
8 word x 4 bit ROM



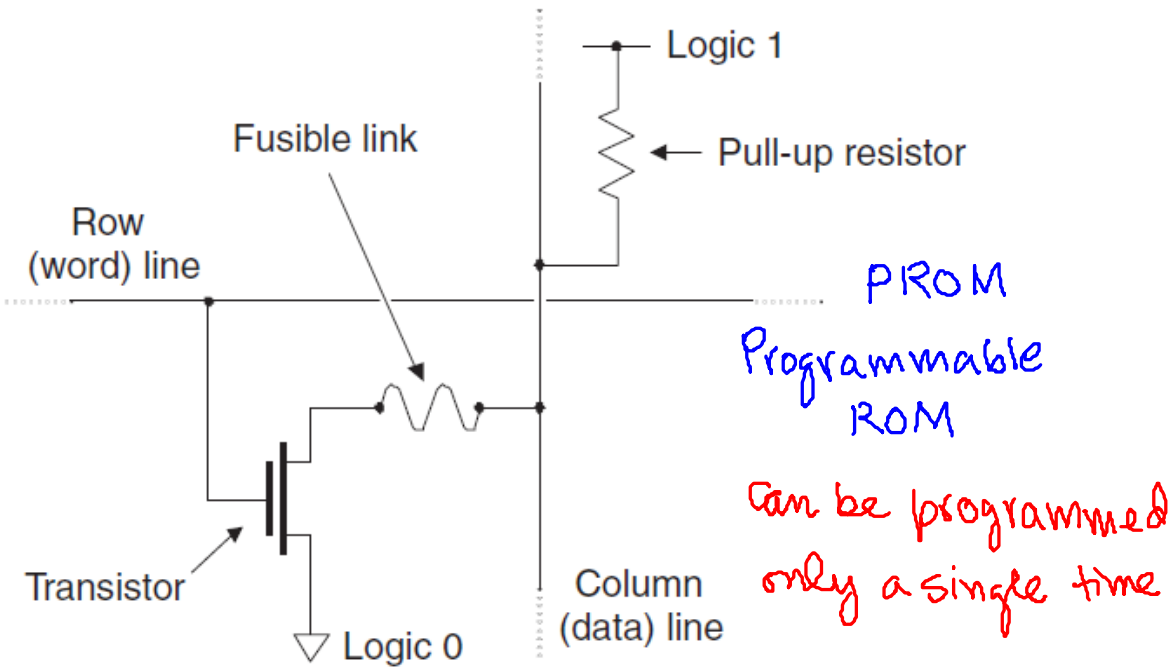
Switching element connected if minterm' is to be included

$$F_0 = (m_0' m_1' m_4' m_6')$$
$$= (m_0 + m_1 + m_4 + m_6)'$$

AND ARRAY

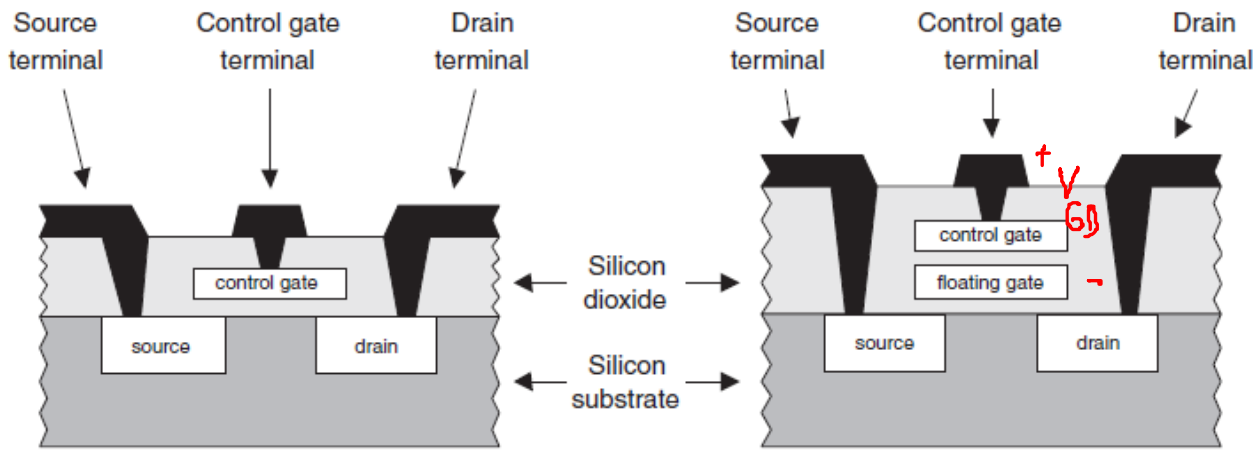


A transistor-based mask-programmed ROM cell



A transistor-and-fusible-link–based PROM cell

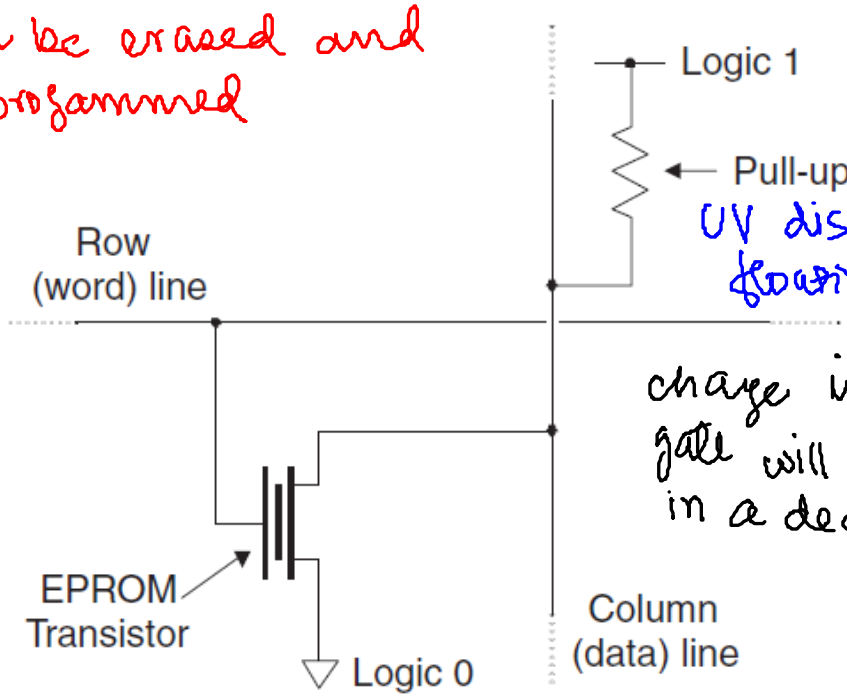
EPROM- Erasable PROM



(a) Standard MOS transistor (b) EPROM transistor

Can be erased and reprogammed

High V_{GD} charges floating gate.



UV discharges floating gate

time to erase ~20 min

charge in floating gate will not dissipate in a decade

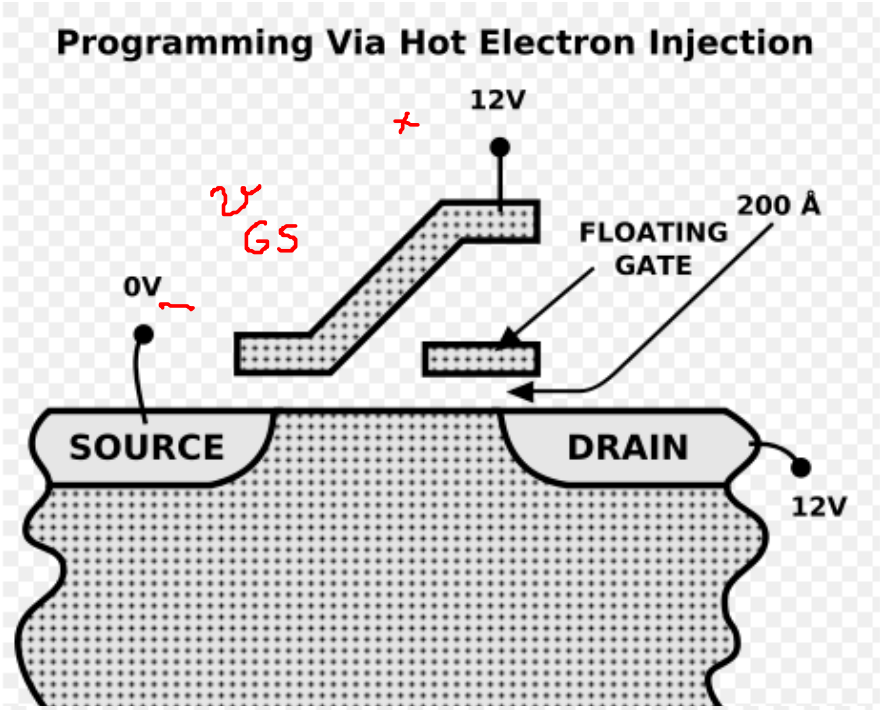
An EPROM transistor-based memory cell.

uncharged floating gate : normal transistor
charged floating gate : no transistor action

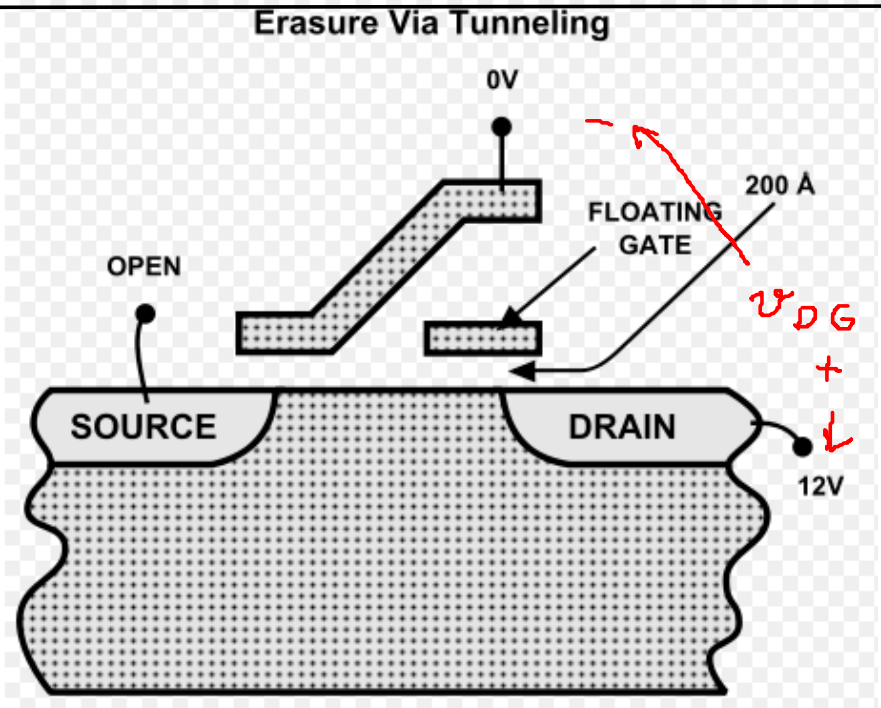
EEPROM Electrically erasable PROM

E^2PROM

FLASH-based technologies



charging the floating gate



uncharging the floating gate