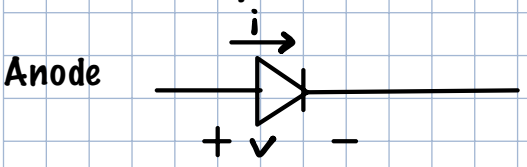
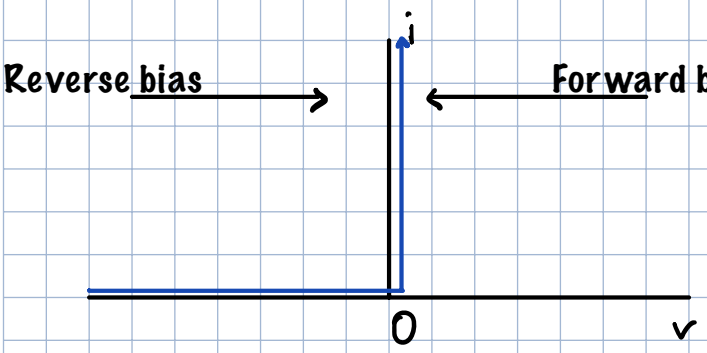
Chapter 9 - ROM, RAM, Programmable Logic Devices

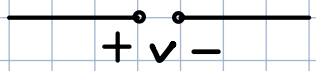
Diode

Electronic component that conducts current primarily in one direction. E.g. p-n junction diode

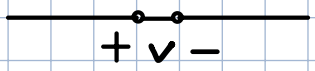


Voltage-current characteristics (ideal)



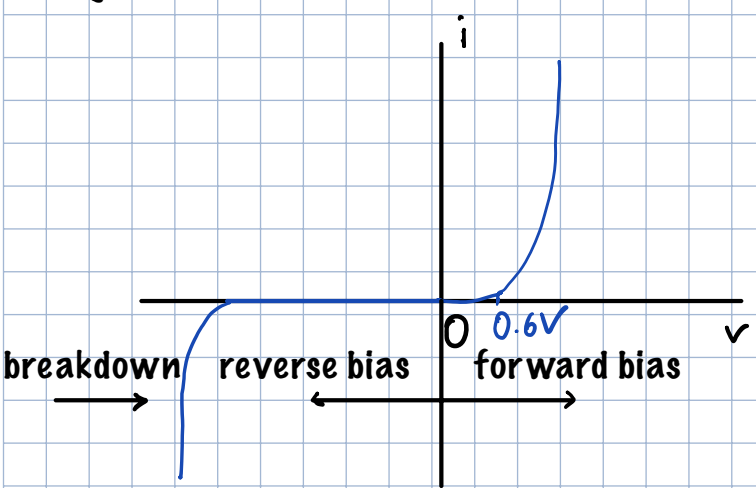


v<0 then i=0



i>0 then v=0

Voltage-current characteristics (real)



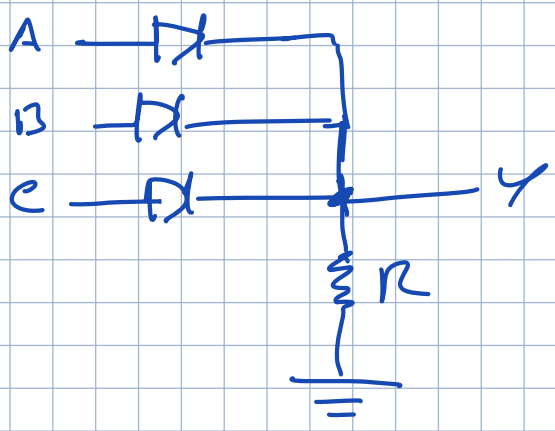
William Shockley diode equation (co-inventor of transistor, 1956 Nobel prize):

(current) $i = I_s * (\exp(v / (n V_T)) - 1)$

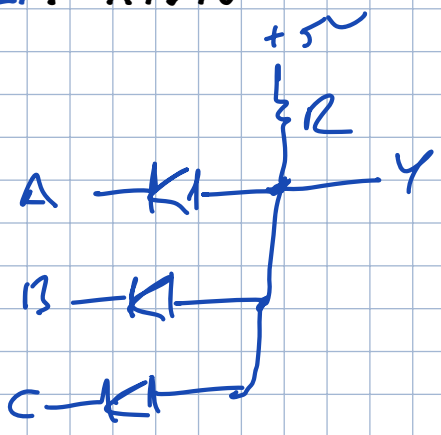
- I_s: saturation current
- n: ideality factor (between 1 and 2)
- V_T: thermal voltage (temperature dependent)

Logic functions using diodes and resistors.

Ex 1: Y = A + B + C



Ex 2: Y = A . B . C

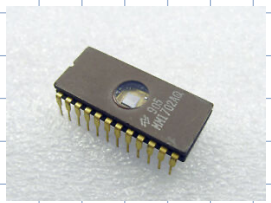


ROM (Read only memory)

- lookup table
- stores binary codes
- used to store firmware, TV remote control, BIOS, etc. Stores binary codes for sequence instructions

Many types:

Masked ROM, Programmable ROM (PROM), UV Erasable PROM (EPROM), Electrically Erasable PROM (EEPROM)

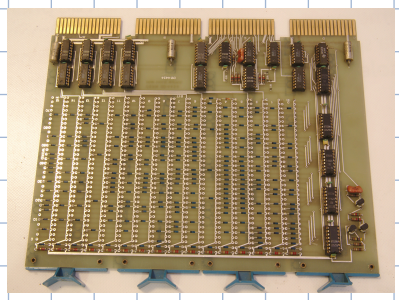


non-volatile storage

Diode ROM

8x4

8 nibble ROM



In a PROM, fuse can be blown by passing a high current for a small time.

This can only be programmed once.

data stored = 8x4 matrix

- Default value for data is "1". Note: horizontal and vertical wires do not intersect!
- Put a diode whenever you want a "0" OR blow a fuse whenever you want a "1".
- Any truth table can be implemented.
- OE: Enables read operation
- CS: Used to select from multiple chips.

Data stored in the ROM above:

$f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$

$D_3 = f(x_1, x_2, x_3)$

x_2 x_1 x_0		$D_3 D_2 D_1 D_0$
000	0	1110
001		1111
010		1110
011	0	1111
100		1100
101	0	1111
110	0	1000
111		1111

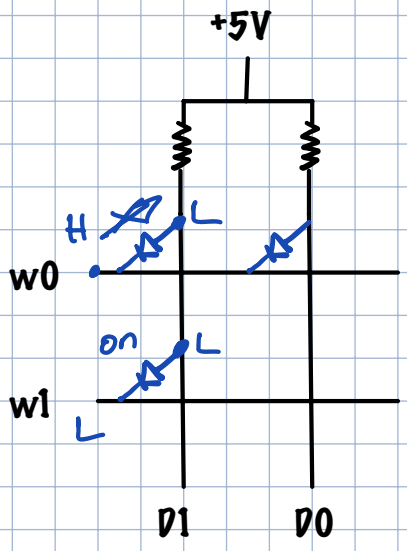
⇒ can implement 4 3-input logic functions.

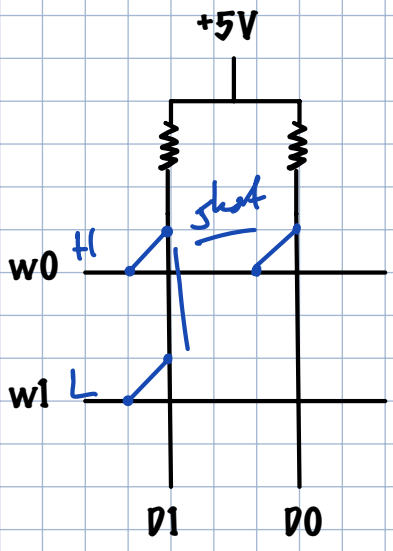
n columns

2^m rows

- $2^m \times n$ ROM can be built using m to 2^m decoder for the address bits
- Any m input n output combinational circuit can be stored in a $2^n \times m$ ROM.

Are diodes necessary?





w1 =low then D1 low, D0 high

Other Types of ROM

EPROM

- Uses MOS transistors instead of diodes
- Programmed with EPROM programmer
- Erase stored data by exposing the chip to UV light for several minutes

EEPROM

8 bits

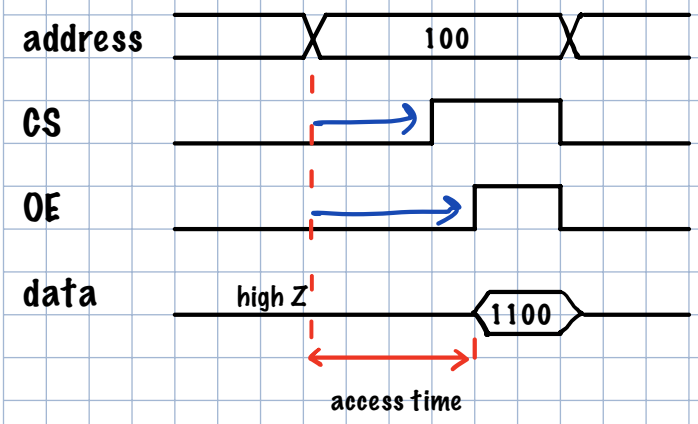
- Allows individual Bytes to be erased or programmed
- Limited life for erasing and reprogramming (reaching a million operations in modern EEPROMs)
- Uses floating-gate MOSFET

Flash Memory (Flash ROM)

- Variation of EEPROM
- EEPROM: erased and programmed at byte level
- Flash: erased and programmed in blocks (faster)
- Used in BIOS (basic input/output system), cell phones, LAN switches, USB flash drive.

ROM Read Timing

How can we read content of address 100?



Exercise: design an FSM to generate correct timing of signals to read from ROM

S_0: idle

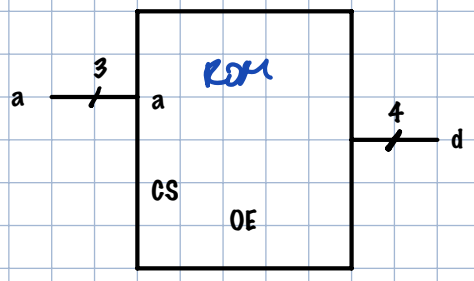
S_1 CLK 1: give address

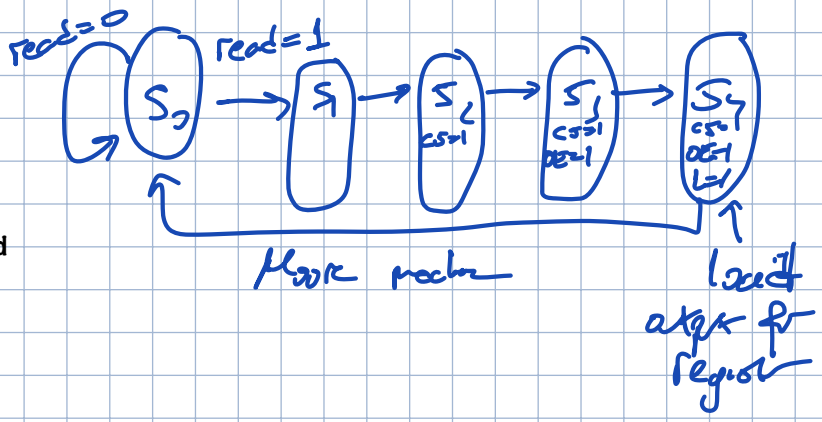
S_2 CLK 2: select chip

S_3 CLK 3: enable output

S_4 CLK : read data into register

symbol for 8 nibble ROM

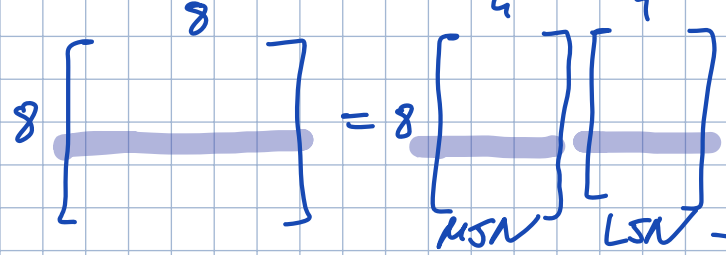


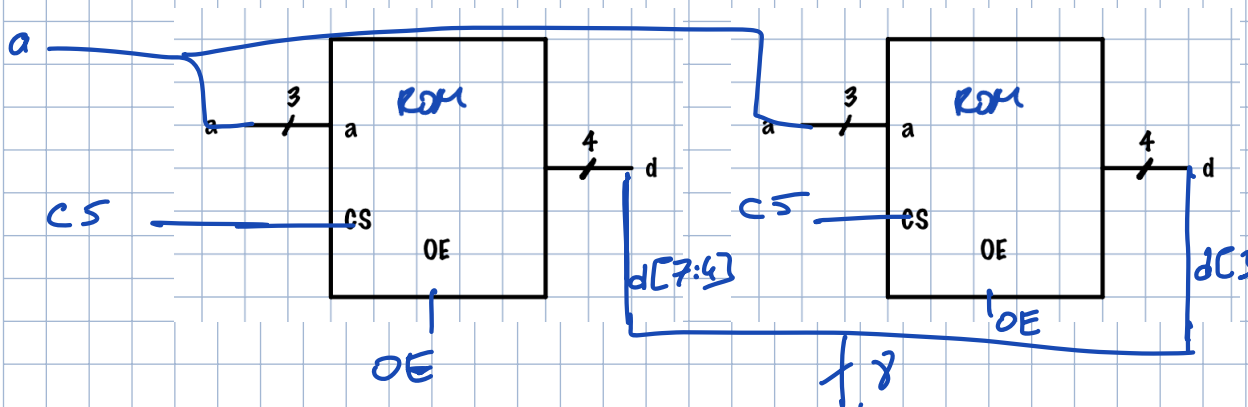


Making wider memory: 8 Byte ROM from 8 nibble ROM

8 Byte ROM: 3 bit address input. CS input, OE input. 8 bit data output.

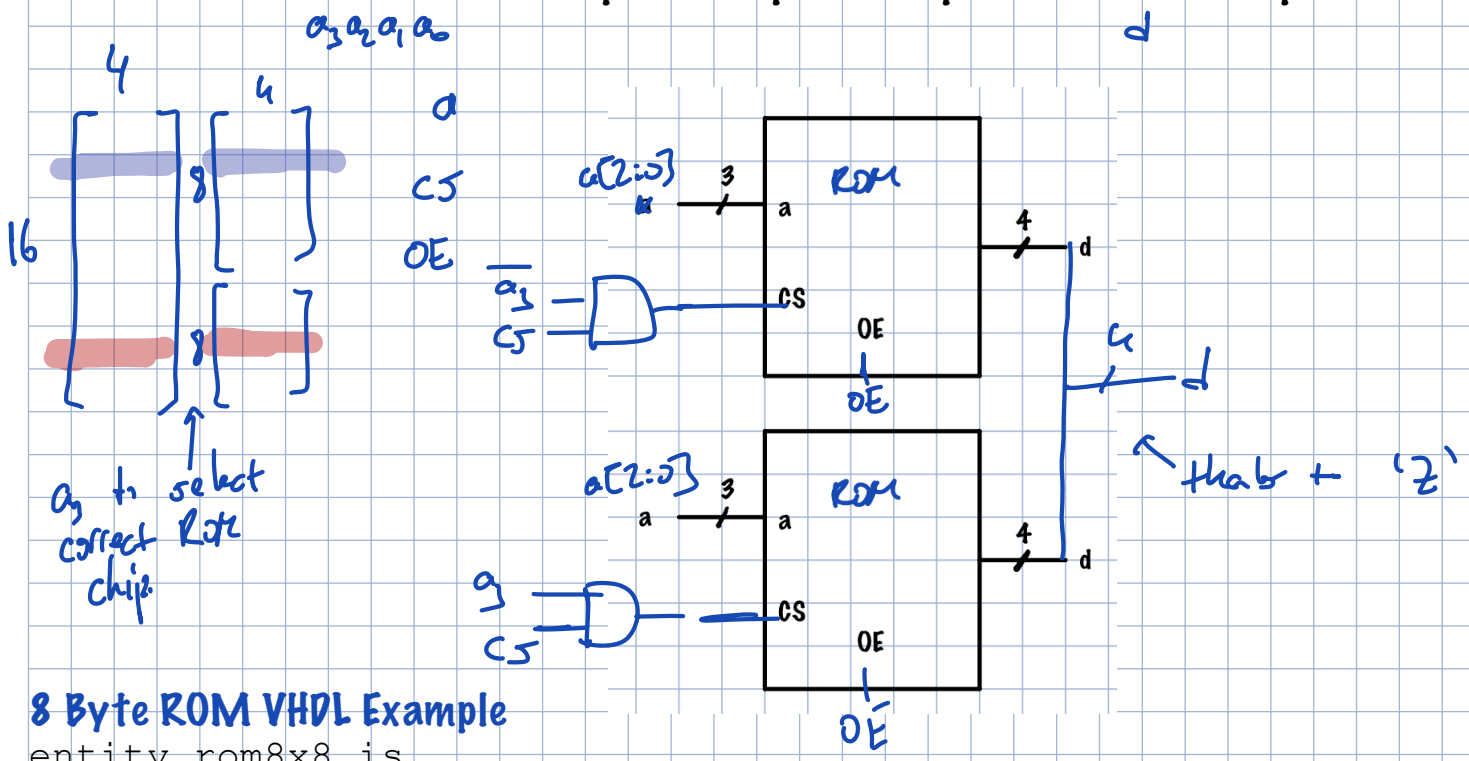
8x8





Making larger memory. 16 nibble ROM from 8 nibble ROM

16 nibble ROM: 4 bit address input. CS input. OE input. 4 bit data output.



8 Byte ROM VHDL Example

```
entity rom8x8 is
    port(address: in integer range 0 to 7;
         data: out std_logic_vector(7 downto 0);
         CS, OE: in std_logic);
end entity;

architecture rom_arch of rom8x8 is
    type rom_array is array (0 to 7) of std_logic_vector(7 downto
0);
    constant rom: rom_array := ("11111011", "00010010",
"10011011", "10010011", "01011011", "00111010", "11111011",
"00010010");
begin

process(address,CS,OE) begin
    data<=(others=>'Z'); --chip not selected
    if CS='1' then
        if OE='1' then --read
            data<=rom(address);
        end if;
    end if;
end process;

end architecture;
```

The `constant` is a data object whose value cannot be changed. Unlike `signal` it does not represent a wire in the circuit.

RAM (Random Access Memory)

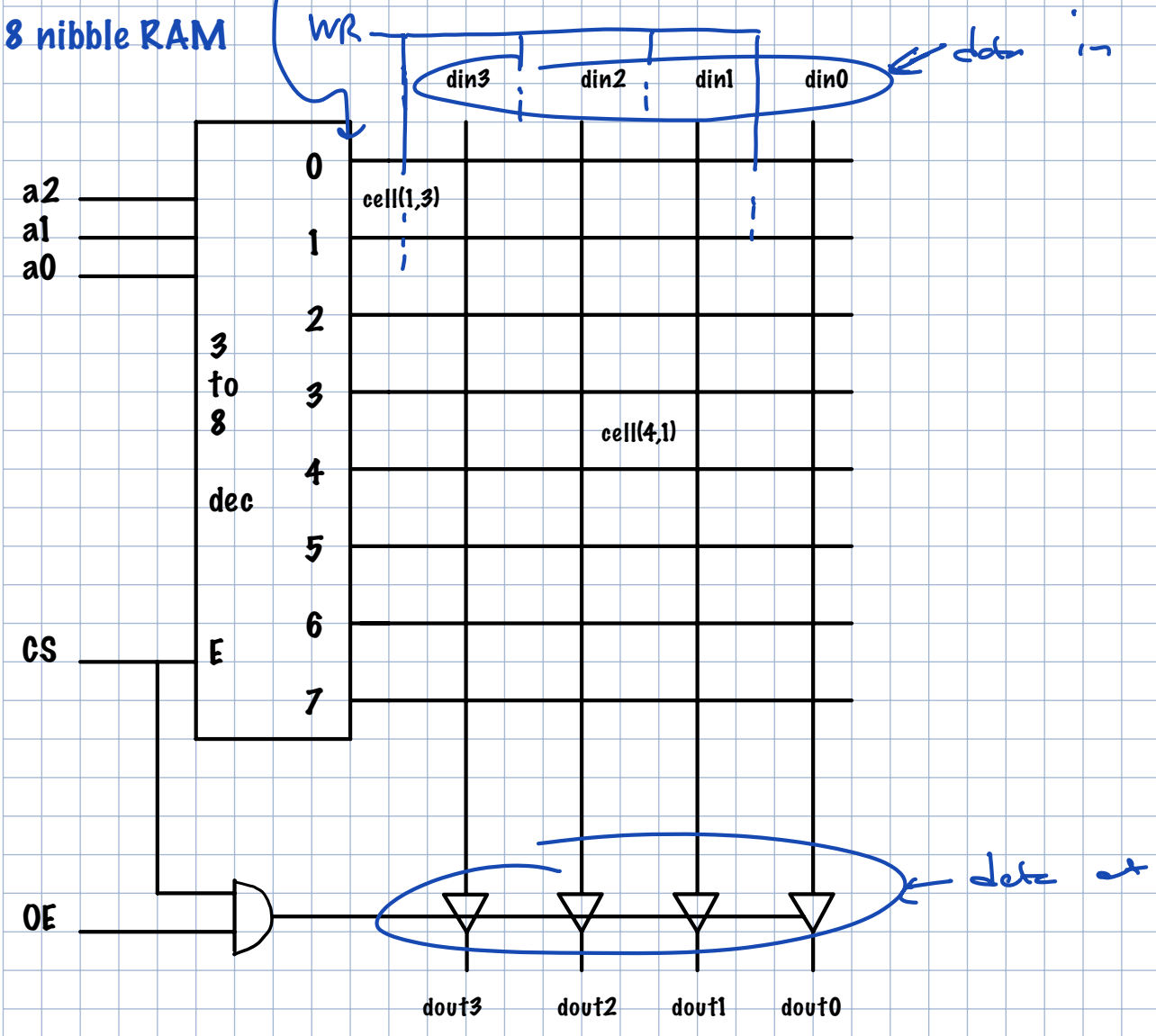
Random access: access time independent of the location (address).

Any storage device with location dependent access time?

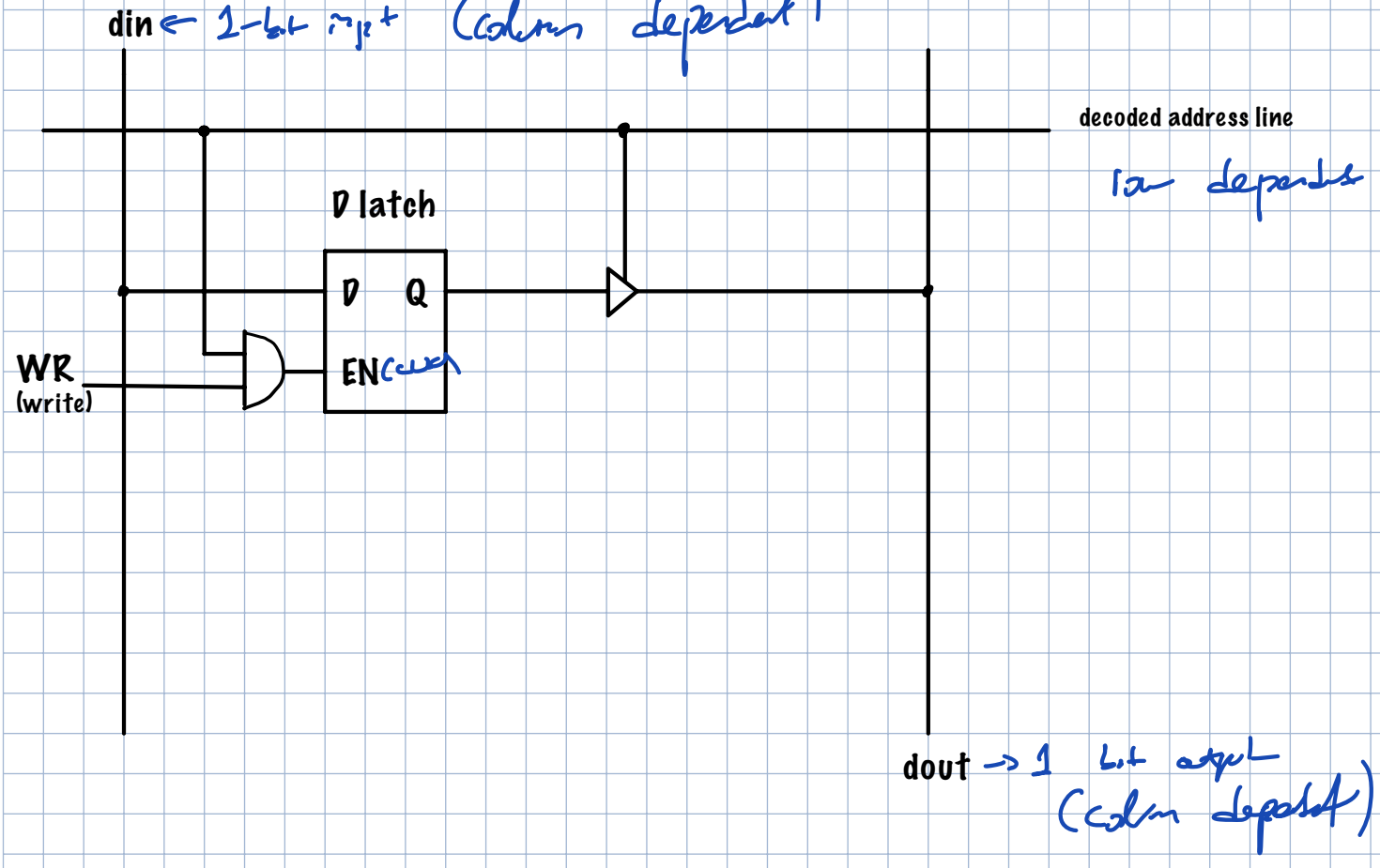
RAM is volatile

Usually operations: read & write

8 nibble RAM



Logic diagram of RAM cell (32 of them in 8×4)



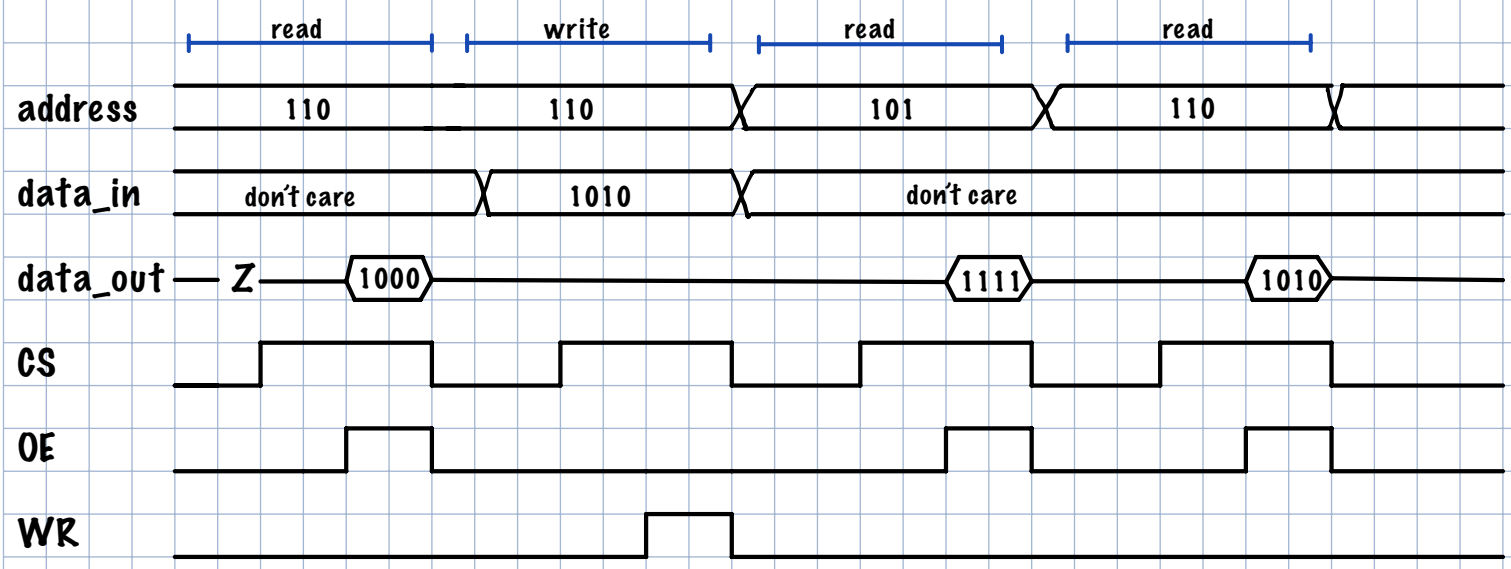
RAM Read and Write Timing

Task:

1. Read address 110
2. Write 1010 to address 110
3. Read address 101
4. Read address 110

Initial contents:

Address	Data
000	1110
001	1111
010	1110
011	1111
100	1100
101	1111
110	1000
111	1111



VHDL code for 64 Byte RAM

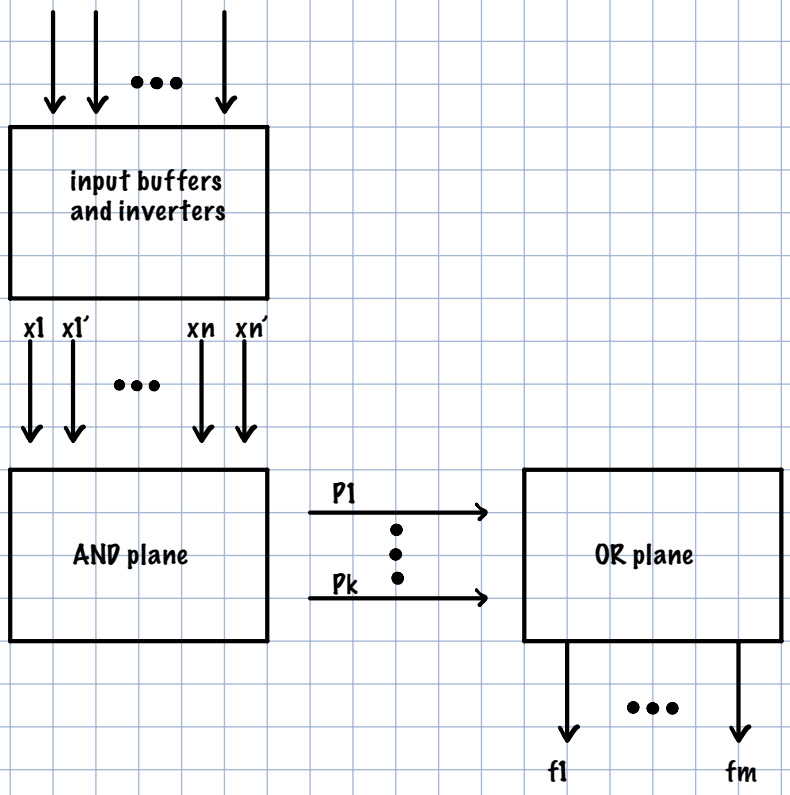
```
use ieee.std_logic_arith.all

entity ram64x8 is
    port(address: in unsigned(5 downto 0);
         data: inout std_logic_vector(7 downto 0);
         WR, CS, OE: in std_logic);
end ram64x8;

architecture ram_arch of ram64x8 is
    type ram_type is array (0 to 63) of std_logic_vector(7 downto
0);
    signal ram1: ram_type;
begin
    process(address,WR,CS,OE) begin
        data<=(others=>'Z'); --chip not selected
        if CS='1' then
            if WR='1' then --write
                ram1(conv_integer(address))<=data;
            end if;
            if WR='0' and OE='1' then --read
                data<=ram1(conv_integer(address));
            else
                data<=(others=>'Z');
            end if;
        end if;
end process;
end ram_arch;
```

conv_integer: **converts its argument to an integer.**

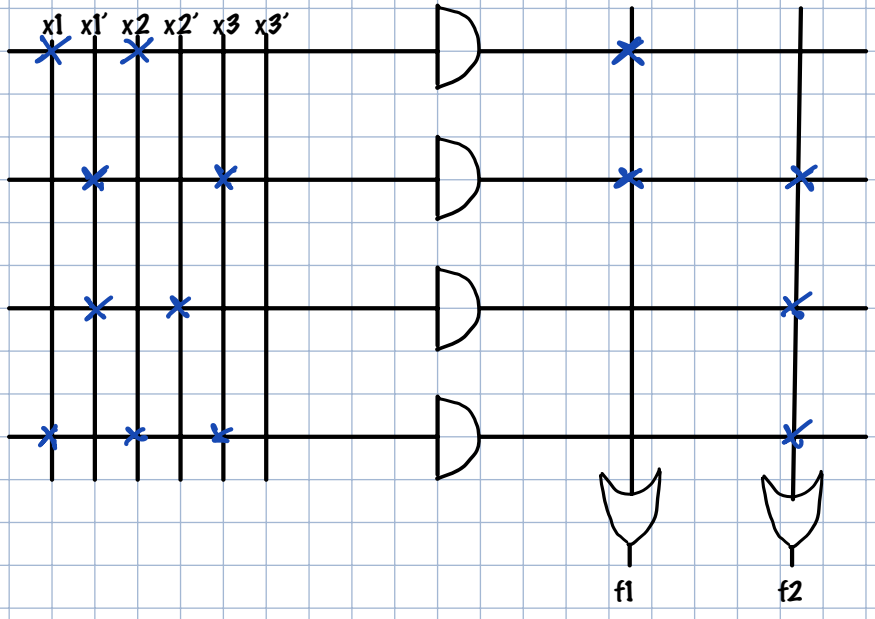
Programmable Logic Device (PLD)

Contains logic gates with programmable switches (introduced in 1970s)

Programmable Logic Array (PLA)



- AND-OR array
- Implements logic functions in sum of products form
- Both AND and OR gates are programmable
- Programing can be made by melting metal fuses (one-time programmable)
- Size of the AND plane constrains the set of functions that can be implemented



Can we implement the following?

f1 = x1x2 + x1′x3

f2 = x1′x3 + x1′x2′ + x1x2x3

What about

f1 = x1x2 + x1′x2x3

f2 = x1′x2′x3′ + x1x2x3 + x1′x2x3′ + x1x3

Not directly
maybe a simplified form
f1 f2

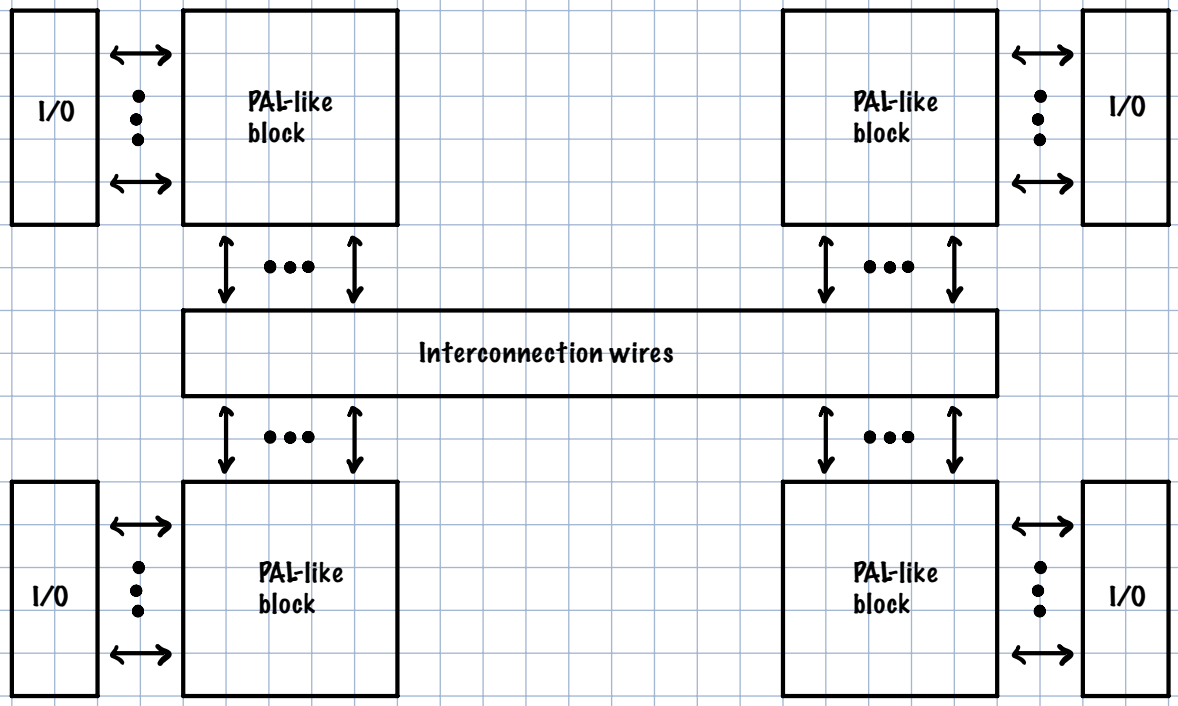
$f_2 = x_1x_3 + \bar{x}_1\bar{x}_3$

Programmable Array Logic (PAL)

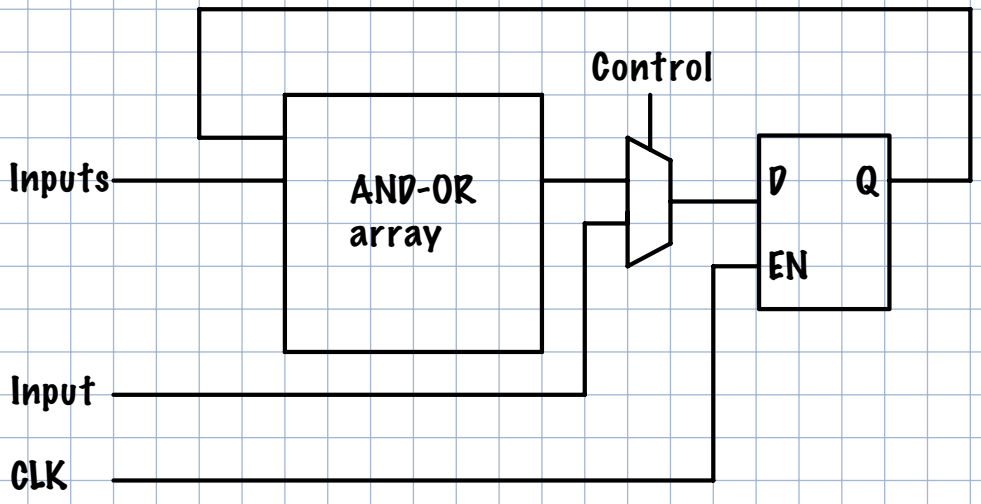
- AND plane programmable, OR plane fixed
- Less expensive than PLA

Complex Programmable Logic Devices (CPLD)

PLA and PAL most of the time limited to 32 inputs+outputs



PAL-like block:



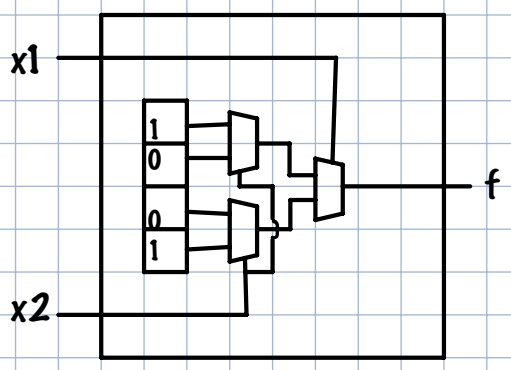
- Large CPLDs have more than 200 PAL-like blocks
- To be programmed in a programming unit, a socket is required to hold the chip
- Sockets are expensive
- Printed circuit board (PCB) that contains CPLD has a small connector to program the CPLD
- JTAG (Joint Test Action Group) port is used to transfer programming information from the computer into the CPLD. This is called in system programming.

FPGA (Field Programmable Gate Array)

- No AND-OR plane
- Can implement millions of gates
- Consists of logic blocks that act like a look up table (LUT)
- Storage cells are made up of SRAM
- FPGA LUTs are volatile

Ex: Basys3. Artix-7 FPGA
-Consists of 5200 slices
-Each slice consists of four 6 input LUT and 8 FFs

Ex: 2 input LUT



x1x2	f
00	1
01	0
10	0
11	1

f = x1 XNOR x2