# King Fahd University of Petroleum & Minerals Computer Engineering Dept

COE 200 – Fundamentals of Computer Engineering

Term 043

Dr. Ashraf S. Hasan Mahmoud

Rm 22-144

Ext. 1724

Email: ashraf@ccse.kfupm.edu.sa



## Definitions

- **Memory**: A collection of cells capable of storing binary information (1s or 0s) – in addition to electronic circuit for storing (writing) and retrieving (reading) information
- **Two Types of Memory**:
  1. Random Access Memory (RAM):
     - Accepts new information for storage to be available later for use
     - *Write/Read* operations
  2. Read Only Memory (ROM): is a programmable logic device (PLD)
     - Programming: specifying information and embedding it within hardware
     - *Read* operation (no write)

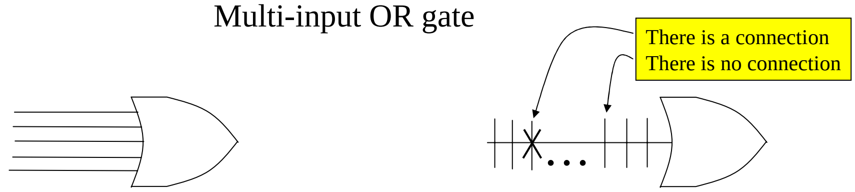
## Definitions (2)

- **<u>Programmable Logic Devices (PLDs)</u>**:
  - ROM
  - Programmable Logic Array (PLA)
  - Programmable Array Logic (PAL) device
  - Complex Programmable Logic Device (CPLD)
  - Field-Programmable Gate Array (FPGA)
- PLD is an integrated circuit with internal logic gates and/or connections that can in some way be changed by a programming process
  - Example: connections can be made with fuses
    - Intact fuse – connection ➔ symbol = "X"
    - Blown fuse - no connection ➔ symbol = "+"



## Used symbol

Multi-input OR gate




conventional symbol array logic symbol

- Most PLD technologies have gates with very high fan-in
- Fuse map: graphic representation of the selected connections

- Refer to section 6.1

# Read-Only Memory (ROM)

- Def: A device in which *"permanent"* binary information is stored
- Block diagram of ROM:

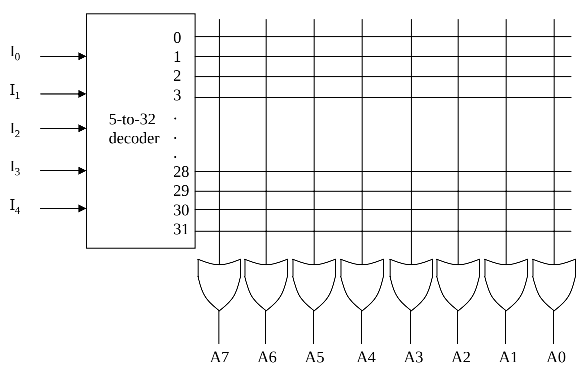

- k inputs → can specify up to $2^k$ words
- Each word is of size n bits
- ROM DOES NOT have a write operation ➔ ROM DOES NOT have data inputs

Word: group of bits stored in one location



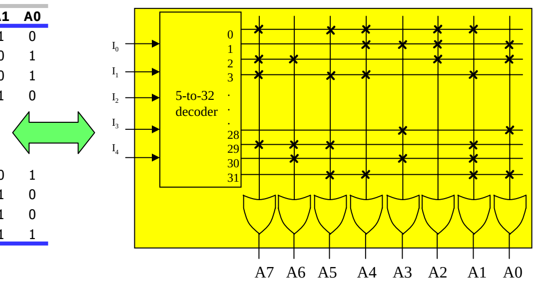
# ROM Internal Logic

- **Note**: The decoder stage produces ALL possible minterms



# Internal Binary Storage of ROM

| Inputs | | | | | Outputs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I4 | I3 | I2 | I1 | I0 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| | | . | | | | | | . | | | | |
| | | . | | | | | | . | | | | |
| | | . | | | | | | . | | | | |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |




- Every ZERO in truth table specifies an OPEN circuit
- Every ONE in truth table specifies a closed circuit
- Example: At address 00011 → The word 10110010 is stored



# Programming Technologies for ROM

1. Mask Technology ➔ ROM
2. Fuses ➔ Programmable ROM or PROM
   - User can blow/connect fuses employing some equipment
3. Erasable floating-gate technology ➔ EPROM
4. Electrically erasable technology ➔ Electrically erasable programmable ROM or EEPROM or $E^2$PROM

## Combinational Circuit Implementation with ROM

- **Problem**: Design a combinational circuit using ROM. The circuit accepts a 3-bit number and generates an output binary number equal to the square of the number.
- **Solution**: Derive truth table:

| Inputs | | | Outputs | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| A2 | A1 | A0 | B5 | B4 | B3 | B2 | B1 | B0 | No |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 16 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 25 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 36 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 49 |

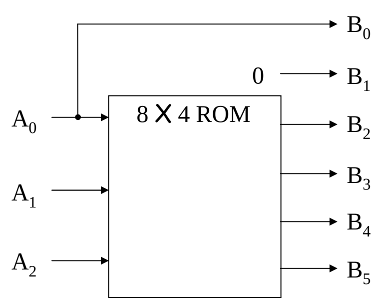


## Combinational Circuit Implementation with ROM – cont'd

- Note that B1 is ALWAYS 0 ➔ no need to generate it using the ROM
- Note that B0 is equal to A0 ➔ no need to generate it using the ROM
- Therefore: The minimum size of ROM needed is $2^3$X4 or 8X4

| Inputs | | | Outputs | | | |
|---|---|---|---|---|---|---|
| A2 | A1 | A0 | B5 | B4 | B3 | B2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 |

ROM truth table – specifies the required connections



## Problem 6-13

- **Problem**: Specify the size of a ROM (number of words and number of bits per word) that will accommodate the truth table for the following combinational circuit components:

(a) An 8-bit adder subtractor with Cin and Cout – assume output is 16-bit wide

(b) A binary multiplier that multiplies two 8-bit numbers

(c) A code converter from a 4-digit BCD number to a binary number



## Problem 6-13 - Solution

- **Solution**:

(a) An 8-bit adder subtractor with Cin and Cout

Inputs to the ROM (address lines):

8 (first number) + (8 second number) + 1 (Cin) + 1 (Add/Subtract) ➔ 18 lines

Hence number of words in ROM is $2^{18}$ = 256K

Size of each word = number of possible functions

= 16 (addition/subtraction) + 1 (Cout)

= 17

Hence ROM size = 256K X 17

## Problem 6-13 - Solution

- **Solution**:

(b) A binary multiplier that multiplies two 8-bit numbers

Inputs to the ROM (address lines):

8 (first number) + (8 second number) ➔ 16 lines

Hence number of words in ROM is $2^{16}$ = 64K

Size of each word = number of possible functions

= 16 (result is 16-bit wide)

Hence ROM size = 64K X 16



## Problem 6-13 - Solution

- **Solution**:

(c) A code converter from a 4-digit BCD number to a binary number

Inputs to the ROM (address lines):

4 (number of BCD digits) X (4 bits/digit) ➔ 16 lines

Hence number of words in ROM is $2^{16}$ = 64K

Size of each word = number of possible functions to represent all 4-digits BCD number

= number of bits required to represent 9999

= 14 (remember $2^{13} \leq 9999 \leq 2^{14}$)

Hence ROM size = 64K X 14

# Problem 6-14

- **Problem**: Tabulate the truth for an 8 X 4 ROM that implements the following four Boolean functions:

A(X,Y,Z) = $\Sigma m(3,6,7)$; B(X,Y,Z) = $\Sigma m(0,1,4,5,6)$

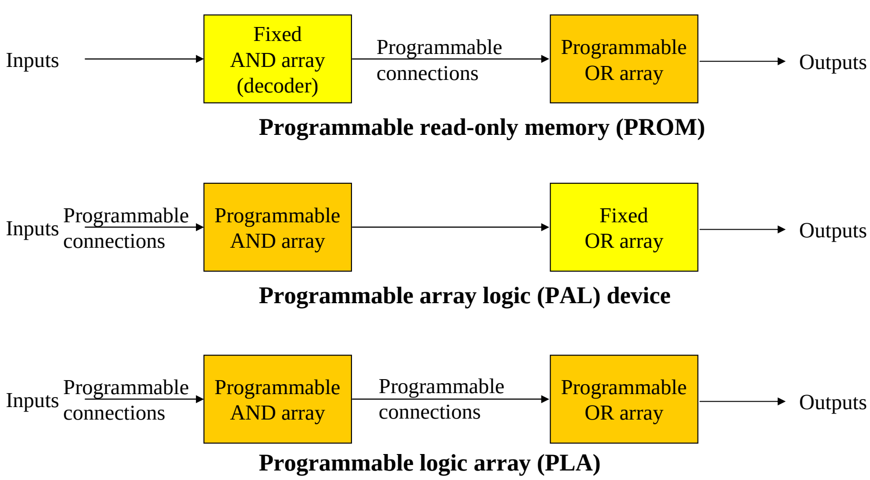
C(X,Y,Z) = $\Sigma m(2,3,4)$; D(X,Y,Z) = $\Sigma m(2,3,4,7)$

- **Solution**:

| Inputs | | | Outputs | | | |
|---|---|---|---|---|---|---|
| X | Y | Z | A | B | C | D |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 |



# Basic Configuration of PLDs



## Programmable Logic Array (PLA)

- Unlike PROM, PLA does NOT provide full decoding of the variables; i.e. it does not generate all the minterms
- The decoder is replaced with an array of AND gates that can be programmed to generate product terms of the input variables
- The terms are then *selectively* (programmable connections) connected to OR gates to provide the sum products for the required Boolean functions



## Example 1:

- Problem: Implement the following functions using a PLA

$$F_1 = AB' + AC + A'BC'; \quad F_2 = (AC + BC)'$$

# Example 1: PLA Programming Table

**PLA programming table:**

Specifying the function form:
T: function does not require inverting
C: function requires inverting

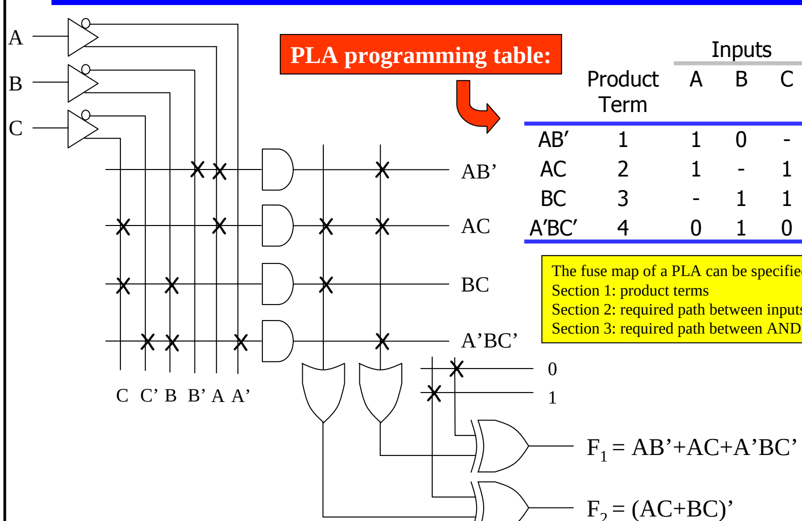
| | Product Term | Inputs A | Inputs B | Inputs C | Outputs (T) F1 | Outputs (C) F2 |
|---|---|---|---|---|---|---|
| AB′ | 1 | 1 | 0 | - | 1 | - |
| AC | 2 | 1 | - | 1 | 1 | 1 |
| BC | 3 | - | 1 | 1 | - | 1 |
| A′BC′ | 4 | 0 | 1 | 0 | 1 | - |

Sections: 1 (product terms), 2 (inputs), 3 (outputs)

The fuse map of a PLA can be specified in this tabular form
Section 1: product terms
Section 2: required path between inputs and AND gates
Section 3: required path between AND and OR gates



# Example 1: PLA with 3 Inputs, 4 Product Terms, and 2 Outputs




**PLA programming table:**

| | Product Term | Inputs A | Inputs B | Inputs C | Outputs (T) F1 | Outputs (C) F2 |
|---|---|---|---|---|---|---|
| AB′ | 1 | 1 | 0 | - | 1 | - |
| AC | 2 | 1 | - | 1 | 1 | 1 |
| BC | 3 | - | 1 | 1 | - | 1 |
| A′BC′ | 4 | 0 | 1 | 0 | 1 | - |

The fuse map of a PLA can be specified in this tabular form
Section 1: product terms
Section 2: required path between inputs and AND gates
Section 3: required path between AND and OR gates

## PLA Internal Logic

- **Notes**:
  - The size of the PLA is specified by:

    no of inputs X no of products X no of outputs
  - In general: for an n inputs, k products, and m outputs PLA:
    - n buffer inverter gates
    - k AND gates
    - m OR gates
    - m XOR gates
    - 2n X k programmable connections (inputs ←→ AND gates)
    - k X m programmable connections (AND gates ←→ OR gates)
    - m programmable connections (for the XOR gates)



## Implementing Combinational Circuits Using PLA

- **Reduce number of distinct products** (i.e. save by reducing number of AND gates):
  - Simplify Boolean function (and its complement) to obtain the least number of products
  - Reducing number of literals in the product is not a saving – since all inputs are already available; However, less is better

# Example 2: Implementing a Combinational Circuit Using a PLA

- Problem: Implement the following two Boolean functions with a PLA:

  $F_1(A,B,C) = \Sigma m(0,1,2,4)$;  $F_2(A,B,C) = \Sigma m(0,5,6,7)$

- **Solution**:

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 |

$F_1 = A'B' + A'C' + B'C'$
$F_1' = AB + AC + BC$

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |

$F_2 = AB + AC + A'B'C'$
$F_2' = A'C + A'B + AB'C'$

The combination that gives a minimum no of product terms is: F1 = (AB + AC + BC)′
F2 = (AB + AC + A′B′C′)

| | Product Term | Inputs A | B | C | Outputs (C) F1 | (T) F2 |
|---|---|---|---|---|---|---|
| AB | 1 | 1 | 1 | - | 1 | 1 |
| AC | 2 | 1 | - | 1 | 1 | 1 |
| BC | 3 | - | 1 | 1 | 1 | - |
| A′B′C′ | 4 | 0 | 0 | 0 | - | 1 |



# Problem 6-16:

- **Problem**: Derive the PLA programming table for the combinational circuit that squares a 3-bit number. Minimize the number of product terms.
- **Solution**:

| Inputs A2 | A1 | A0 | Outputs B5 | B4 | B3 | B2 | B1 | B0 | No |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 16 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 25 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 36 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 49 |

B0 = Z
B1 = 0

| Inputs A2 | A1 | A0 | Outputs B5 | B4 | B3 | B2 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 |

# Problem 6-16: Solution

- **Solution**:

| Inputs | | | Outputs | | | |
|---|---|---|---|---|---|---|
| A2 | A1 | A0 | B5 | B4 | B3 | B2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 |

| $A_2$ \ $A_1A_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |

$B_5 = A_2A_1$
$B_5' = A_2' + A_1'$

| $A_2$ \ $A_1A_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

$B_4 = A_2A_1' + A_2A_0$
$B_4' = A_2' + A_1A_0'$

| $A_2$ \ $A_1A_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 |

$B_2 = A_1A_0'$
$B_2' = A_1' + A_0'$

| $A_2$ \ $A_1A_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |

$B_3 = A_2A_1'A_0 + A_2'A_1A_0$
$B_3' = A_2'A_1' + A_1'A_0' + A_2A_1 + A_1A_0'$



# Problem 6-16: Solution

- **Solution**:

$B_2 = A_1A_0'$
$B_2' = A_1' + A_0'$

$B_3 = A_2A_1'A_0 + A_2'A_1A_0$
$B_3' = A_2'A_1' + A_1'A_0' + A_2A_1 + A_1A_0'$

$B_4 = A_2A_1' + A_2A_0$
$B_4' = A_2' + A_1A_0'$

$B_5 = A_2A_1$
$B_5' = A_2' + A_1'$

Minimum number of products = 5
B0 = A0
B1 = 0

PLA programming Table

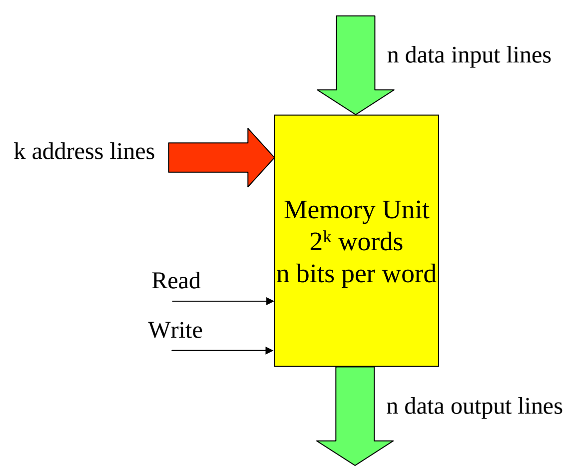
| | Product Term | Inputs | | | Outputs | | | |
|---|---|---|---|---|---|---|---|---|
| | | A2 | A1 | A0 | (T) B5 | (C) B4 | (T) B3 | (T) B2 |
| $A_2'A_1A_0$ | 1 | 0 | 1 | 1 | - | - | 1 | - |
| $A_2A_1'A_0$ | 2 | 1 | 0 | 1 | - | - | 1 | - |
| $A_2A_1$ | 3 | 1 | 1 | - | 1 | - | - | - |
| $A_1A_0'$ | 4 | - | 1 | 0 | - | 1 | - | 1 |
| $A_2'$ | 5 | 0 | - | - | - | 1 | - | - |

## Random Access Memory (RAM)

- Def: A device in which binary information is stored and can be transferred to and from any location - with the access taking the same time regardless of the location
- To compare:
  - Serial Memory: the access time depends on the location of the info – e.g. magnetic disk, tape, etc
- Binary info is stored in words – group of bits in one location
  - A group of size eight is called BYTE



## Block Diagram of RAM



## Read/Write Operation of RAM

- Write Operation:
  1. Apply the binary address of the desired word to the address lines
  2. Apply the data bits that must be stored in memory to the data input lines
  3. Activate the Write input
- Read Operation:
  1. Apply the binary address of the desired word to the address lines
  2. Activate the Read input



## RAM – Control Inputs to a Memory Chip

- Chip select (CS): to enable the particular RAM chip or chips containing the word to be accessed ➔ Memory Enable

| Inputs | | |
|---|---|---|
| Chip Select (CS) | Read/Write' (R/W') | Operation |
| 0 | x | None |
| 1 | 0 | Write to selected word |
| 1 | 1 | Read from selected word |

## Properties of RAM

- Statistic RAM (SRAM):
  - Internal latches that store the binary information. The stored information remains valid as long as power is applied
- Dynamic RAM (DRAM):
  - Stores the binary information in the form of an electric charge on capacitors
  - Must be periodically recharged – *refreshing* ; ALL words are read and re-writting to restore the decaying charge
  - Less power consumption compared to SRAM
  - SRAM is easier to use
- Volatile Memory: info is lost when power is off (e.g. RAM)
- Nonvolatile Memory: info is retained even if power is off (e.g. magnetic disk, ROM, etc)



## Problems of GREAT Interest

- Problems: 6-11, 6-12, 6-13$^{s}$, 6-14$^{s}$, 6-15, 6-16, 6-17