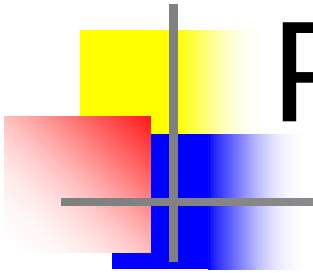
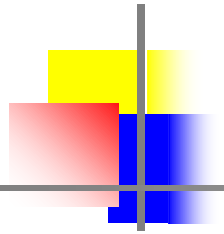


Ch. 07. Memory and Programmable Logic

2019. 5. 20.

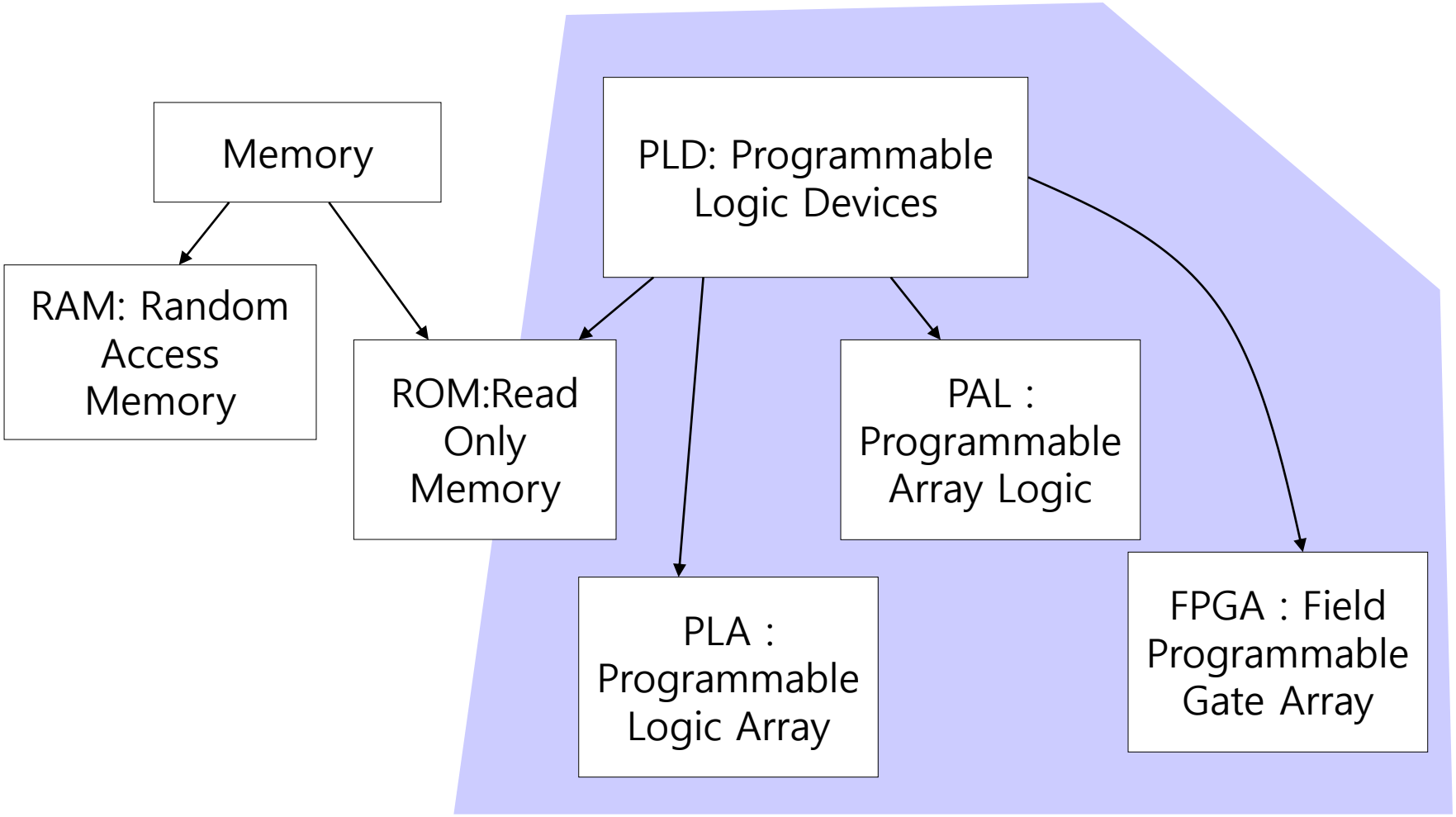
K-S. Sohn



Contents

- Memory unit
  - RAM, ROM
- Programmable logic devices
- Combinational PLDs
  - PROM, PAL, PLA
- Sequential PLDs
  - SPLD, CPLD, FPGA

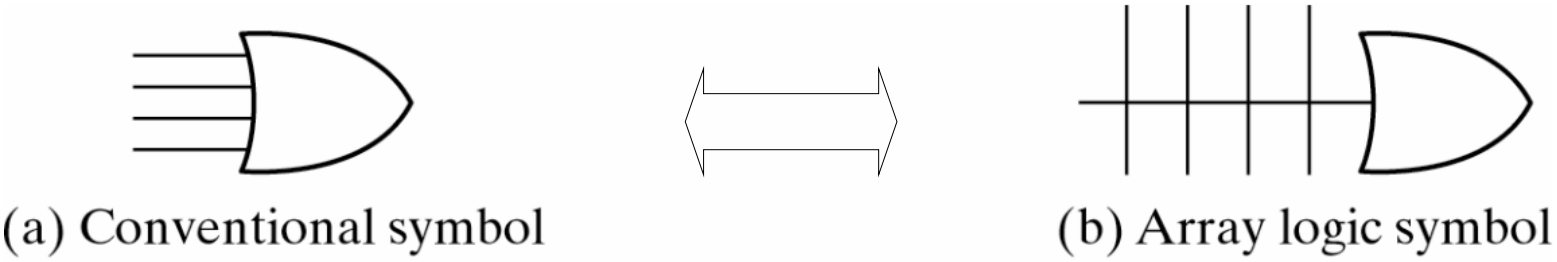
Introduction



Icons for Array Logic Diagram

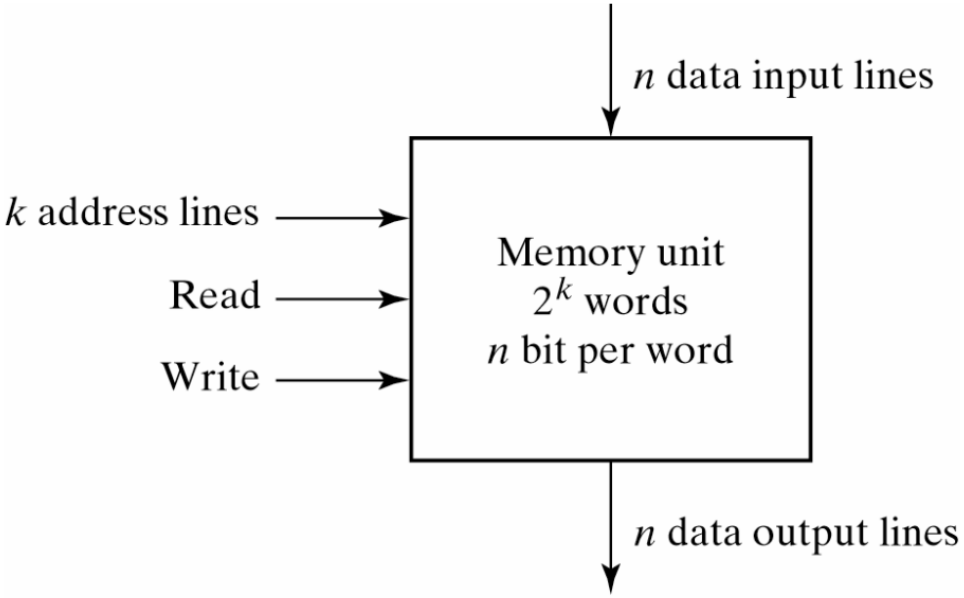
- Objectives
  - To represent the complex internal logic of programmable logic devices in a concise form.

- Examples



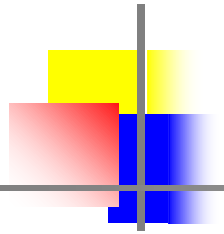
Block Diagram of Memory Unit

- Memory Unit
  - An electronic device that writes/reads n-bit word located by k-bit address
- Word
  - The unit of data written/read to/from the memory unit
  - c.f. : byte, octet



Structure of Memory Unit

Memory address		Memory content
Binary	decimal	
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
⋮	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101



Operations of Memory Units

- Procedure of WRITE operation
  - Apply the address
  - Apply the data
  - Activate the '*Write*' input line
- Procedure of READ operation
  - Apply the address
  - Activate the '*Read*' input line

Control Inputs to Memory Unit

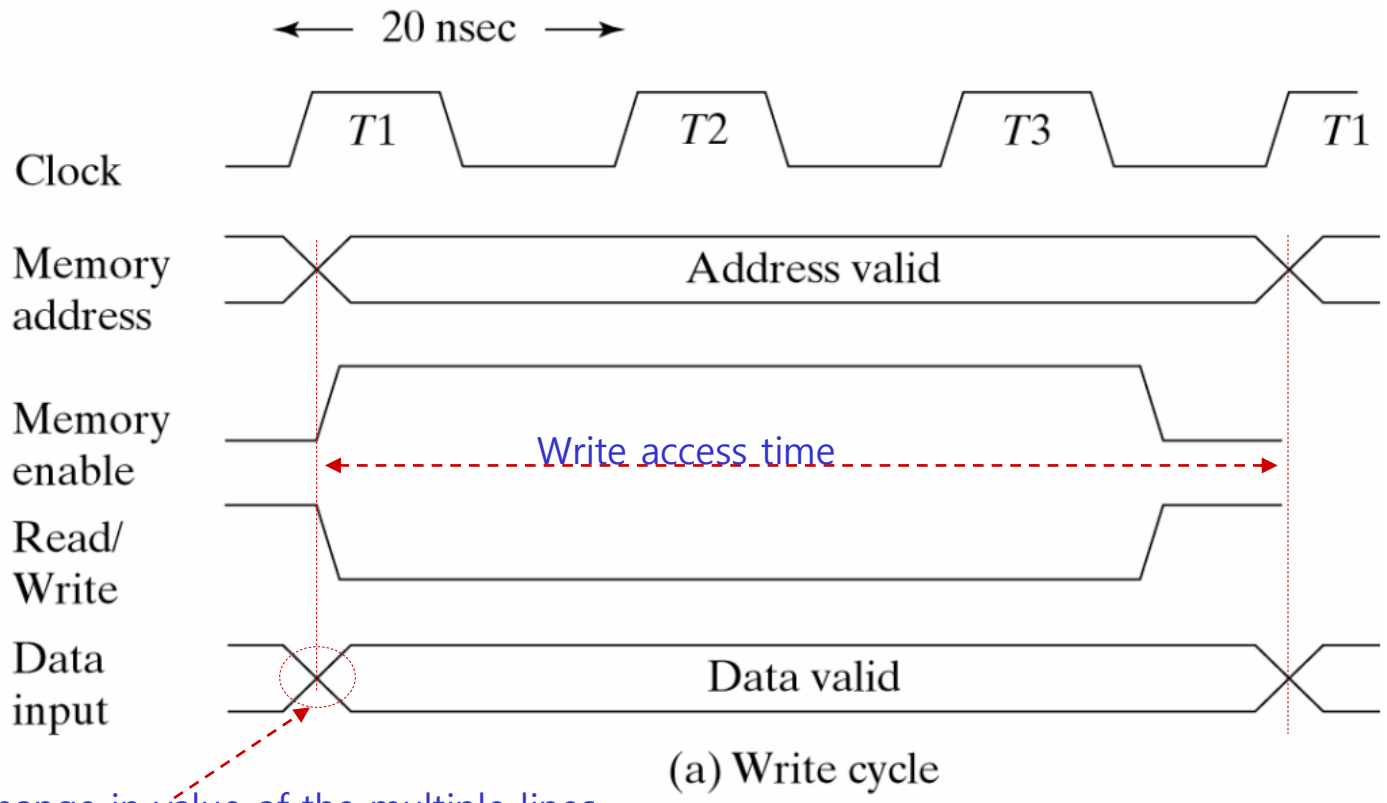
- Control inputs
  - Memory Enable
  - Read/Write

Table 7-1
Control Inputs to Memory Chip

Memory Enable	Read/Write	Memory Operation
0	X	None
1	0	Write to selected word
1	1	Read from selected word

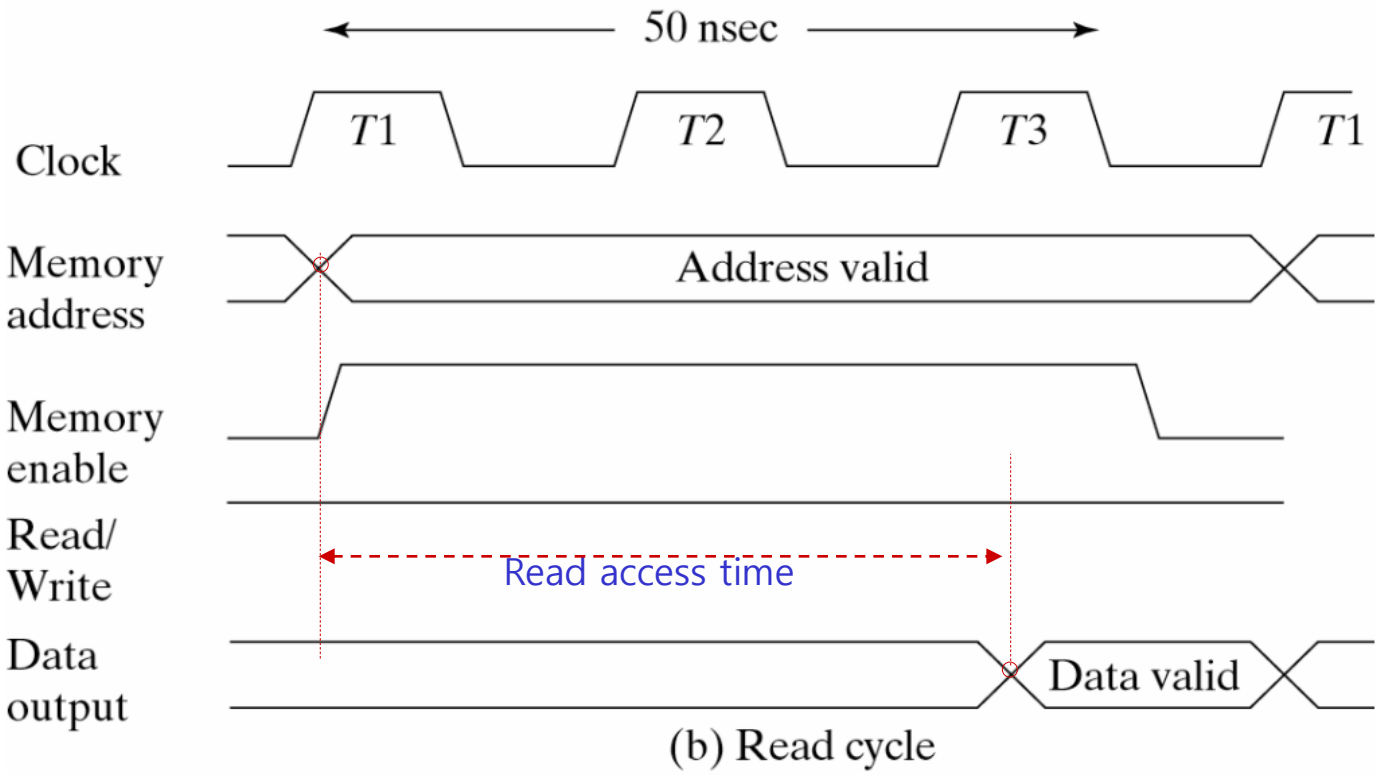
Memory Operation in Timing Diagram

- WRITE operation

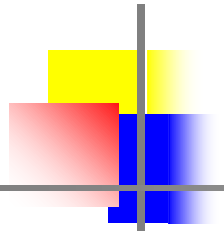


Memory Operation in Timing Diagram

- READ operation



(b) Read cycle

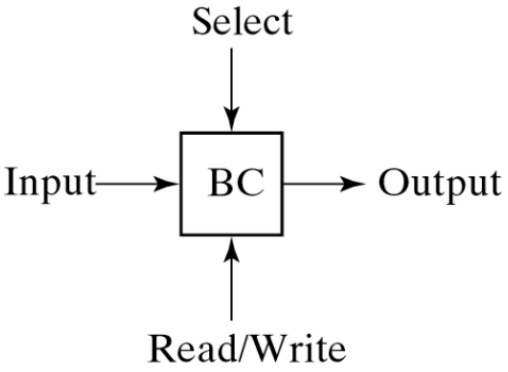


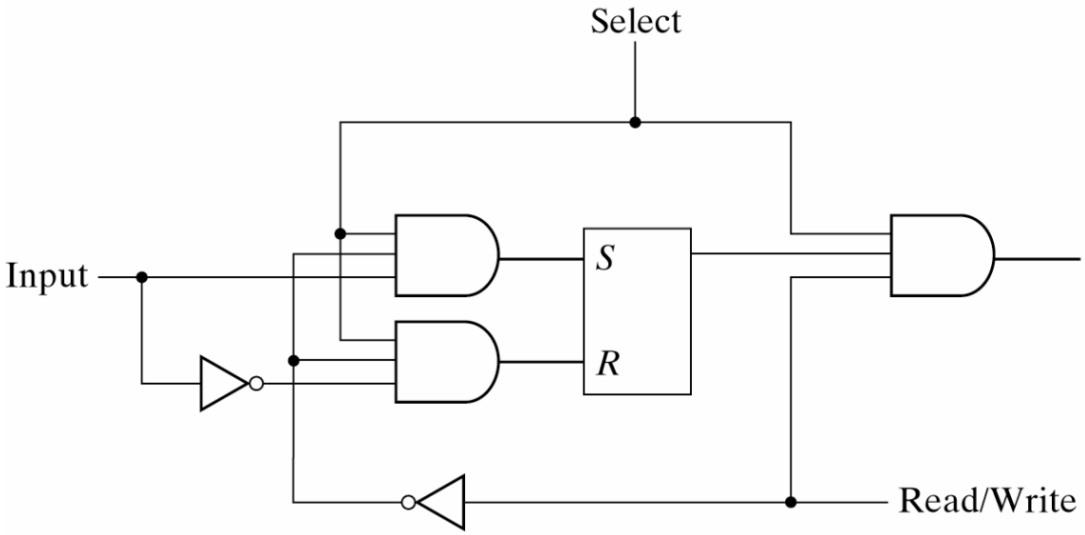
Types of Memories

- SRAM : static RAM
  - Fast
- DRAM : dynamic RAM
  - Slower but smaller space required (1/4 of SRAM)
  - Lower power discipation
  - MOSFET and capacitor
  - Refreshing (recharging the internal capacitor)
- ROM : read only memory
  - Nonvolatile memory

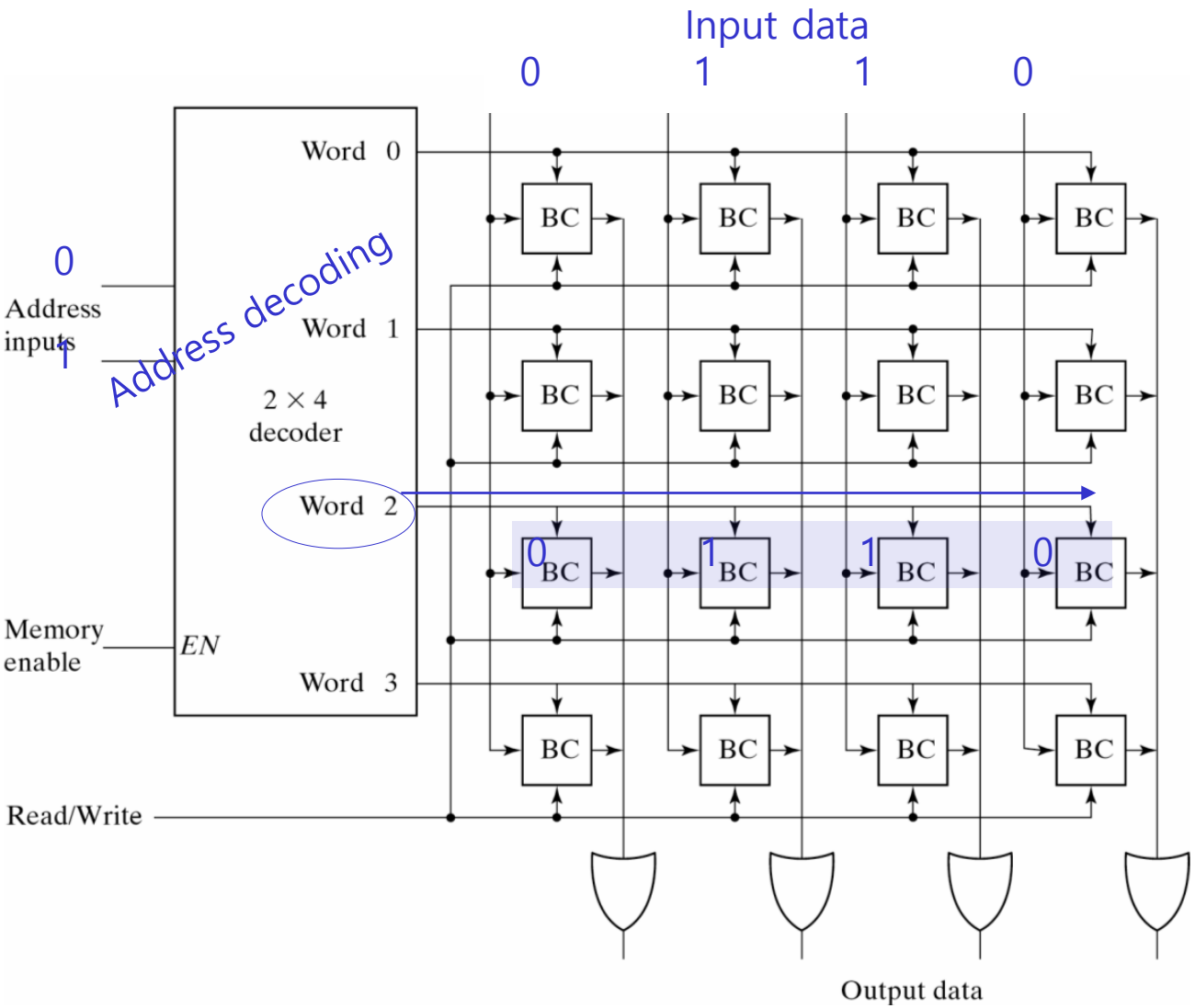
Memory Cell

- Block diagram
  - Select : generated by address decoding
  - Input, Output : 1-bit data lines
- Logic diagram



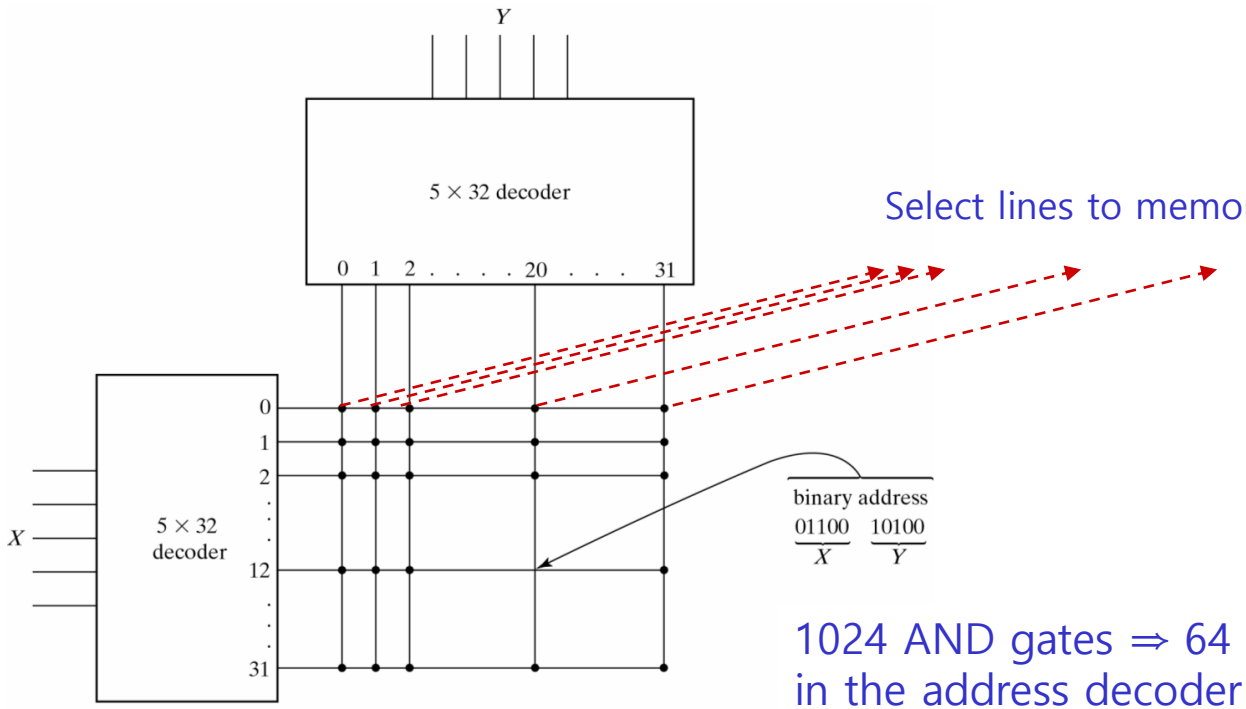


Memory Construction



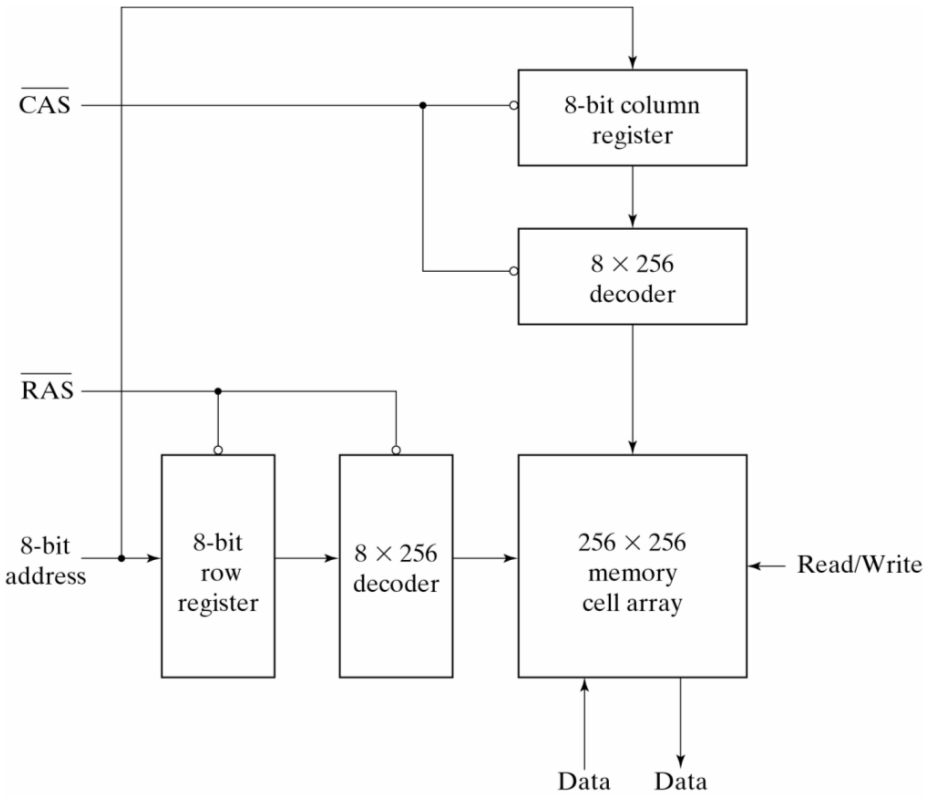
Coincident Address Decoding

- Objectives
  - To reduce the size of the address decoder by space multiplexing

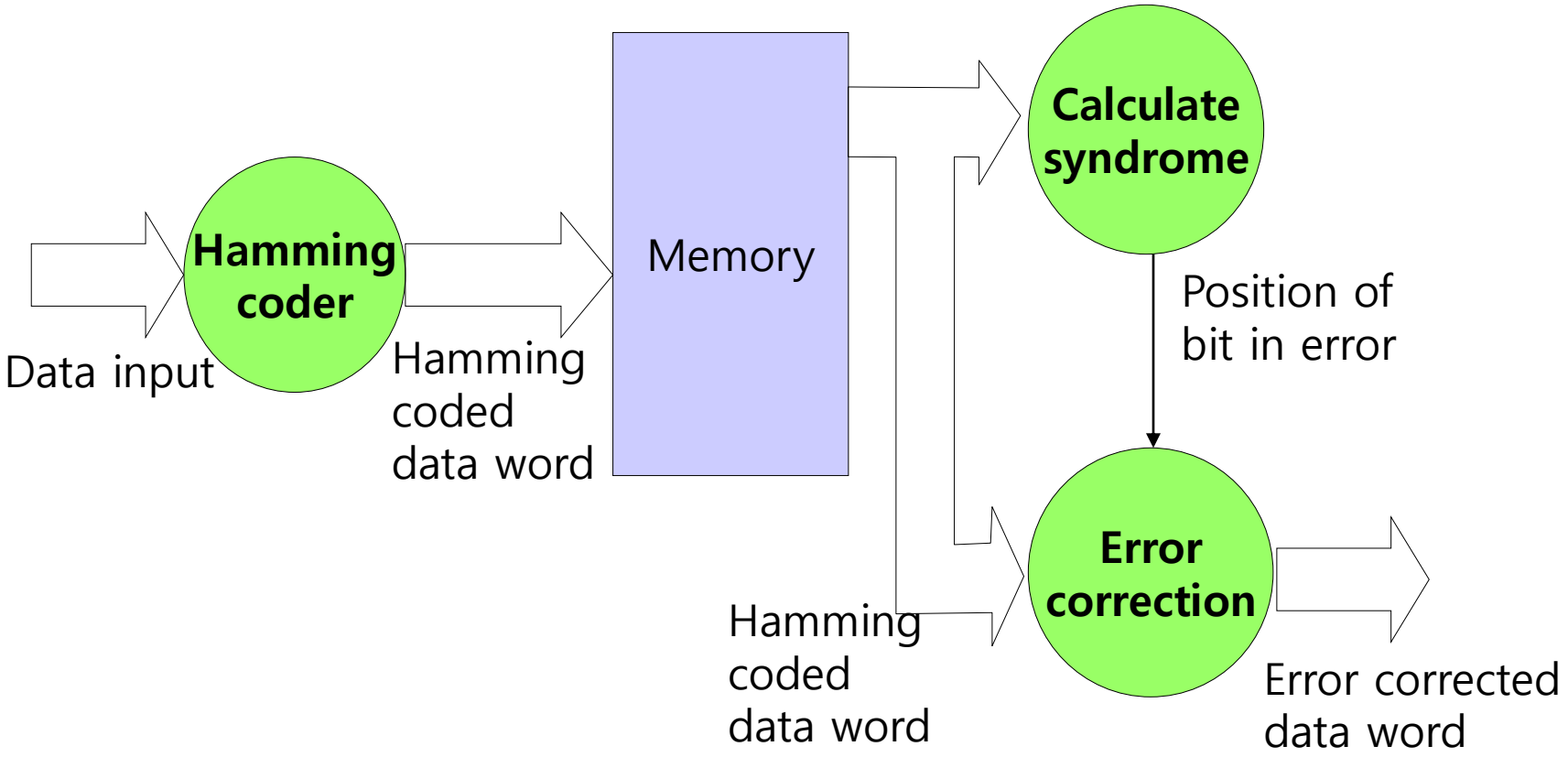


Address Multiplexing

- Objectives
  - To reduce the size of address decoder
  - To reduce the number of pins in the IC package
- Operational description
  - Addressing with a time lag
  - Time division multiplexing version of the coincident addressing
  - CAS(column address strobe) first, and then, RAS(row address strobe)



Error Detection And Correction



Bit Error Syndrome

- Assumption
  - Bit position number : 1, 2, …$n+k$, where n is the number of data bits and k is the number of parity bits.
- Syndrome
  - A binary number designating the position of the bit being in error
  - $S=S_{k-1}...S_0 \Rightarrow$ A syndrome S is a k-bit binary number (Ex. $S=3_{(10)}$ indicates that the 3rd bit of the checked word is in error.)
  - If $S_i=0$, all bits whose positions are numbered by the binary number in which the i-th significant bit is 1 are correct
  - If $S_i=1$, one or more such a bits are in error.

Hamming Code

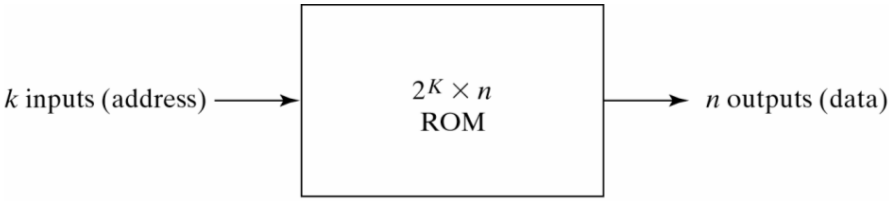
- Assumption
  - k parity bits(p_1,p_2,…,p_{k-1}) are added to an n-bits data word
- Coding scheme
  - The bit positions are numbered in sequence from 1 to $n+k$
  - Those positions numbered as a power of 2 are reserved for the parity bits
  - p_i = XOR of bits that the i-th significant bits of their position number is 1
- Example
  - 8-bit data word : 11000100
  - Hamming code of 1100011 : **00**1**1**100**1**0100

Syndrome of Hamming Code

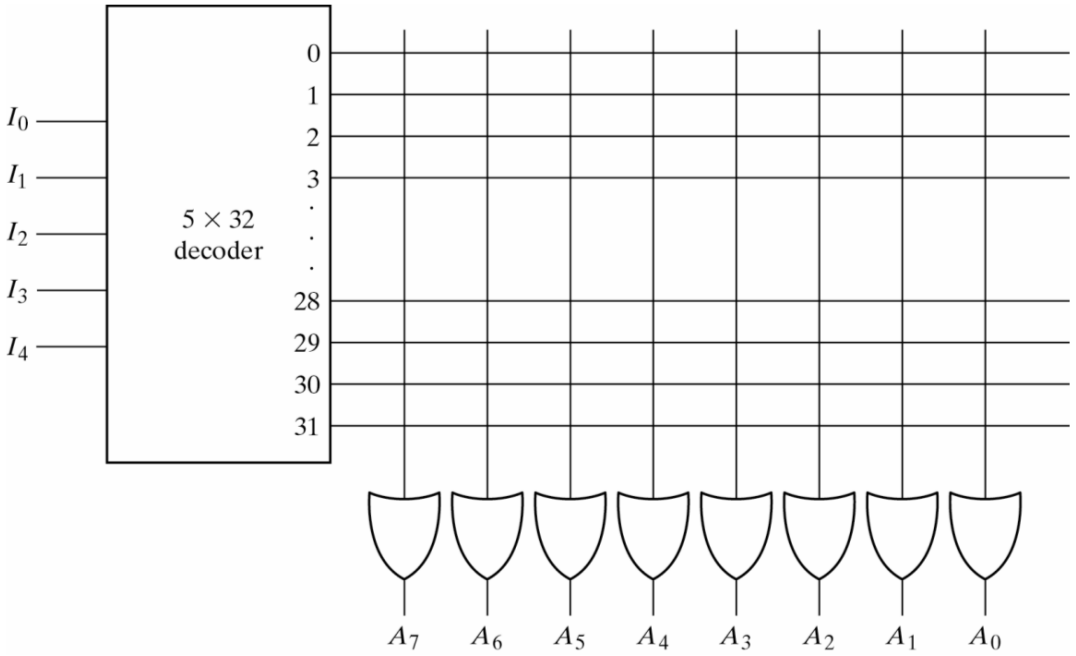
- Evaluation of syndrome from a Hamming Coded memory word
  - S_i = XOR of bits p_i and the data bits that the i-th significant bit of their position number is 1.
- Example
  - k=4, n=8, hamming coded word=00111001**0**100
  - S_0=XOR of bits(1,3,5,7,9,11)
  - S_1=XOR of bits(2,3,6,7,10,11)
  - S_2=XOR of bits(4,5,6,7,12)
  - S_3=XOR of bits(8,9,10,11,12)

Read Only Memory (ROM)

- Permanent storage
  - Block diagram of ROM



  - Internal logic of ROM (Ex.: 32x8 ROM)



ROM

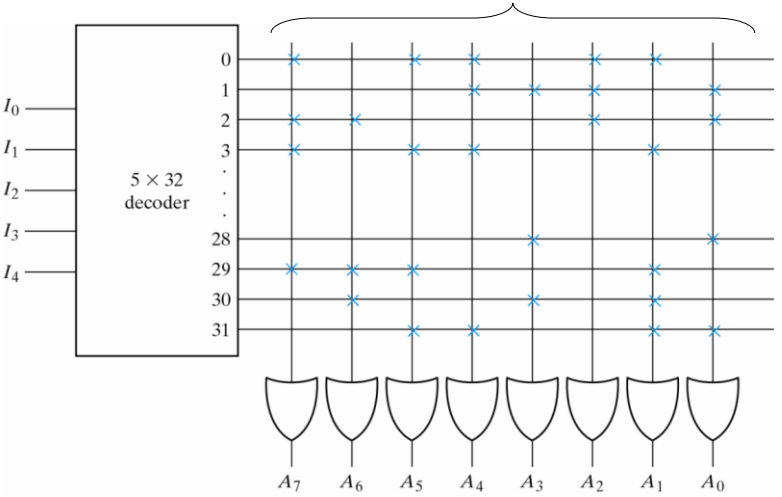
- Programming the ROM as the permanent storage

ROM Truth Table (Partial)

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

List of 32 8-bit words

ROM programmer
generating high voltage signal to blow out fuse of the contact point corresponding to '0' value.



Combinational Circuit Implementation with ROM

- Reinterpretation of the ROM truth table

ROM Truth Table (Partial)

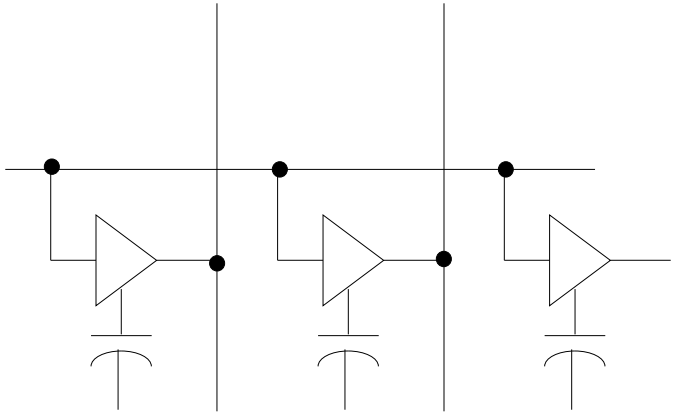
Inputs						Outputs							
I4	I3	I2	I1	I0		A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	m_0	1	0	1	1	0	1	1	0
0	0	0	0	1	m_1	0	0	0	1	1	1	0	1
0	0	0	1	0	m_2	1	1	0	0	0	1	0	1
0	0	0	1	1	m_3	1	0	1	1	0	0	1	0
		⋮			…			⋮					
1	1	1	0	0	m_{28}	0	0	0	0	1	0	0	1
1	1	1	0	1	m_{29}	1	1	1	0	0	0	1	0
1	1	1	1	0	m_{30}	0	1	0	0	1	0	1	0
1	1	1	1	1	m_{31}	0	0	1	1	0	0	1	1

List of minterm

$A_7=\Sigma(0,2,3,\ldots,29)$

$A_1=\Sigma(0,3,\ldots,29,30,31)$

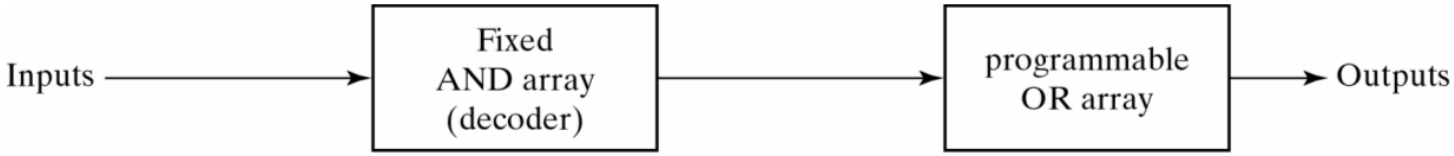
Types of ROM



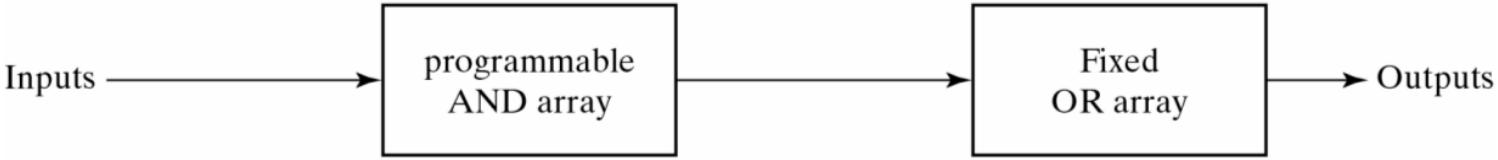
- Mask ROM
- PROM(Programmable ROM)
- EPROM(Erasable PROM)
  - Erase = to discharge the internal floating gate with U.V. light
- EEPROM(Electrically EPROM)
  - Erase = to discharge the internal floating gate electrically

Types of Combinational PLDs

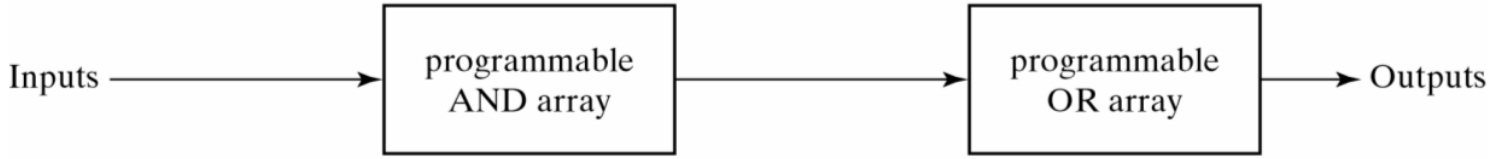
- PROM



- PAL (Programmable Array Logic)



- PLA(Programmable Logic Array)

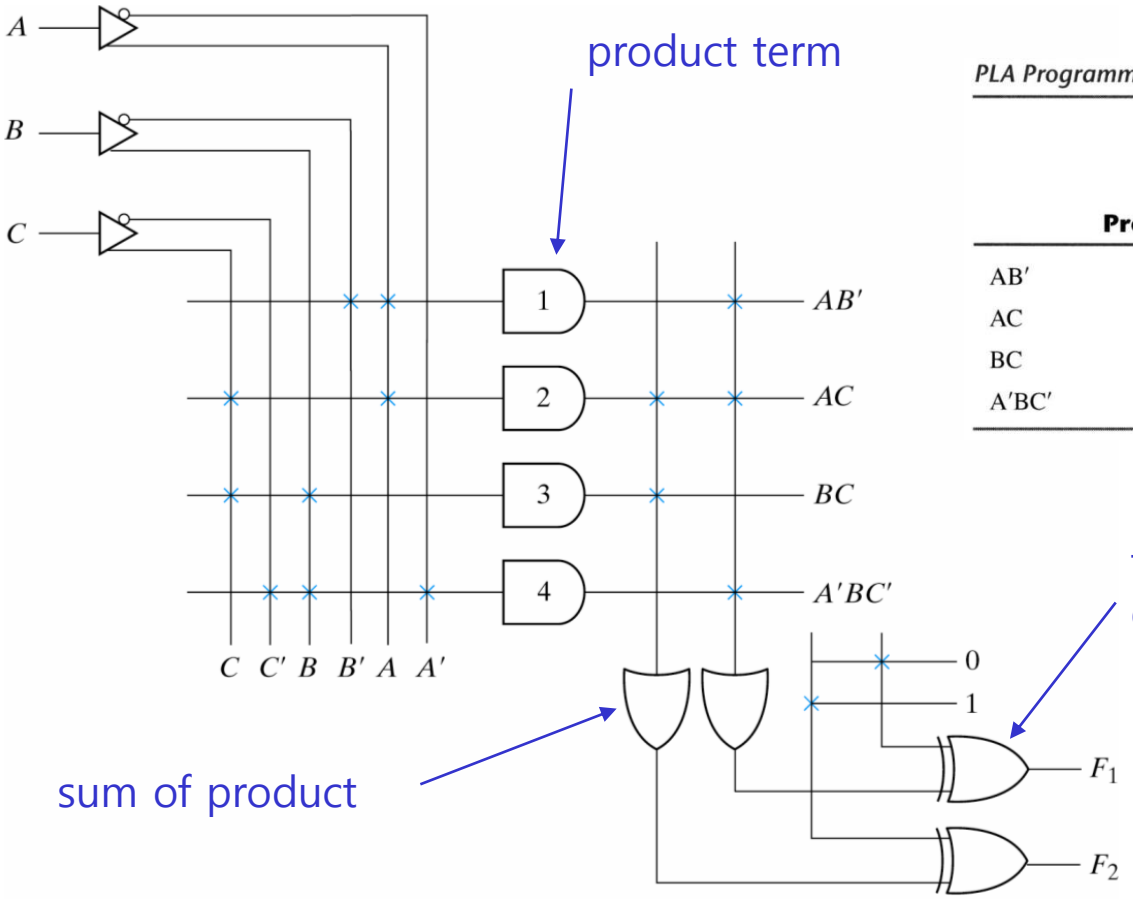


PLA(Programmable Logic Array)

- PROM and PLA

PROM	PLA
Decoder	Array of AND gates
Full minterms	Partial minterms
Full decoding of the variables	Partial decoding of the variables

Internal Logic of a PLA



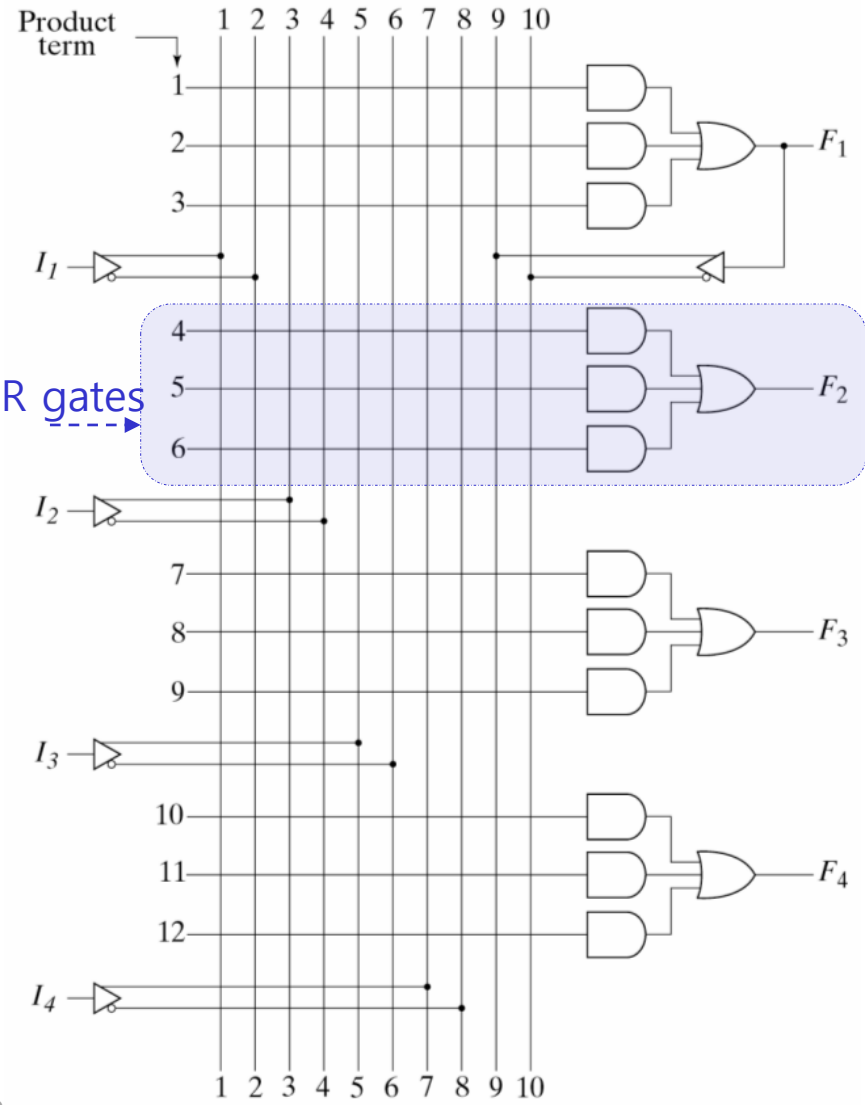
PLA Programming Table

		Inputs			Outputs (T)	(C)
	Product Term	**A**	**B**	**C**	F_1	F_2
AB′	1	1	0	–	1	–
AC	2	1	–	1	1	1
BC	3	–	1	1	–	1
A′BC′	4	0	1	0	1	–

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

$$F_2 = (AC + BC)'$$

PAL(Programming Logic Array)



Section
3 AND, 1 OR gates

Array of sections

☞ Section : A unit of programmable logic

PAL Programming

- Requirement
  - Boolean functions for required combinational circuits

$$w(A,B,C,D) = \sum(2,12,13)$$
$$x(A,B,C,D) = \sum(7,8,9,10,11,12,13,14,15)$$
$$y(A,B,C,D) = \sum(0,2,3,4,5,6,7,8,10,11,15)$$
$$z(A,B,C,D) = \sum(1,2,8,12,13)$$

- Simplifying the circuit

$$w = ABC' + A'B'CD'$$
$$x = A + BCD$$
$$y = A'B + CD + B'D'$$
$$z = ABC' + A'B'CD' + AC'D' + A'B'C'D$$
$$= w + AC'D' + A'B'C'D$$

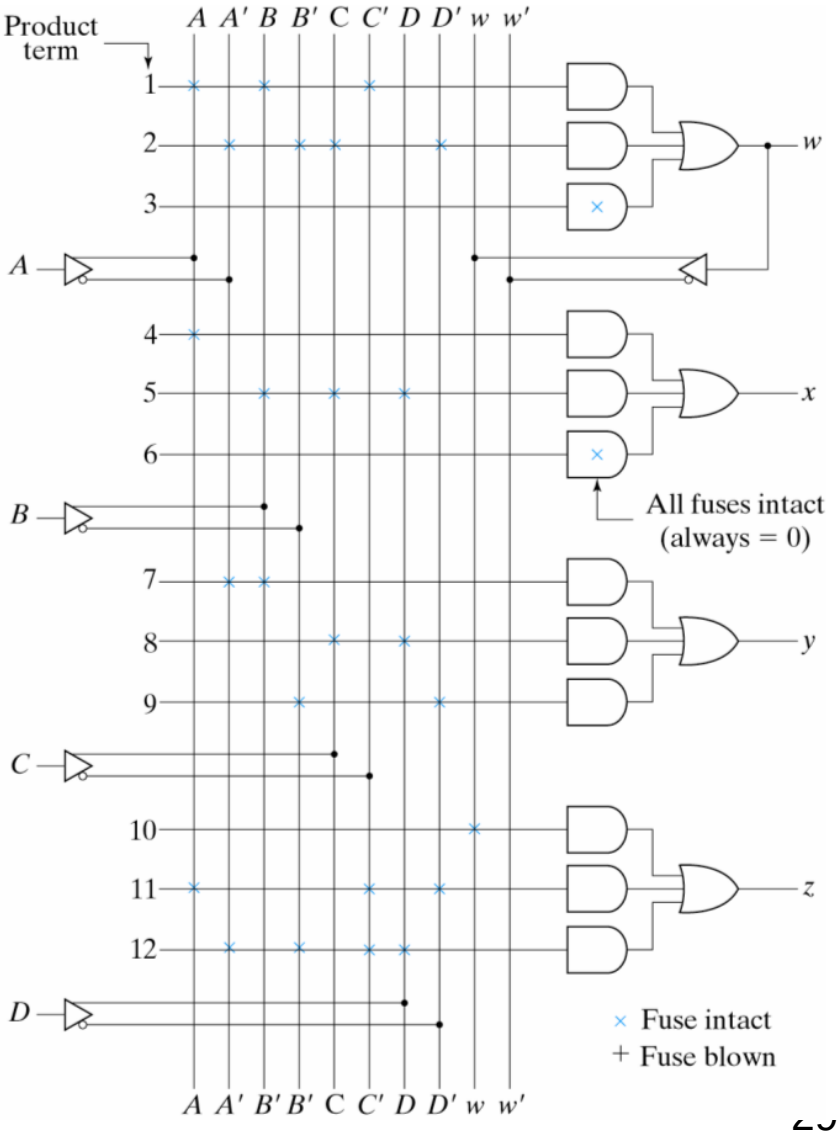
- PAL programming table

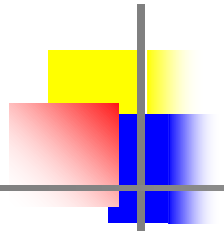
Product Term	AND Inputs A	B	C	D	W	Outputs
1	1	1	0	–	–	$w = ABC'$
2	0	0	1	0	–	$+ A'B'CD'$
3	–	–	–	–	–	
4	1	–	–	–	–	$x = A$
5	–	1	1	1	–	$+ BCD$
6	–	–	–	–	–	
7	0	1	–	–	–	$y = A'B$
8	–	–	1	1	–	$+ CD$
9	–	0	–	0	–	$+ B'D'$
10	–	–	–	–	1	$z = w$
11	1	–	0	0	–	$+ AC'D'$
12	0	0	0	1	–	$+ A'B'C'D$

PAL Programming

- PAL fuse map

$$w = ABC' + A'B'CD'$$
$$x = A + BCD$$
$$y = A'B + CD + B'D'$$
$$z = ABC' + A'B'CD' + AC'D' + A'B'C'D$$
$$= w + AC'D' + A'B'C'D$$

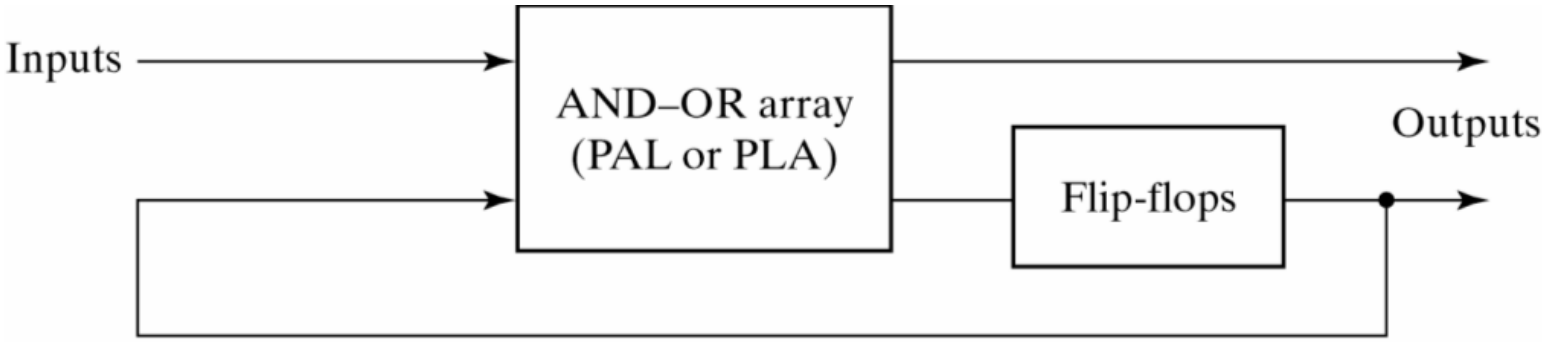




Sequential PLD

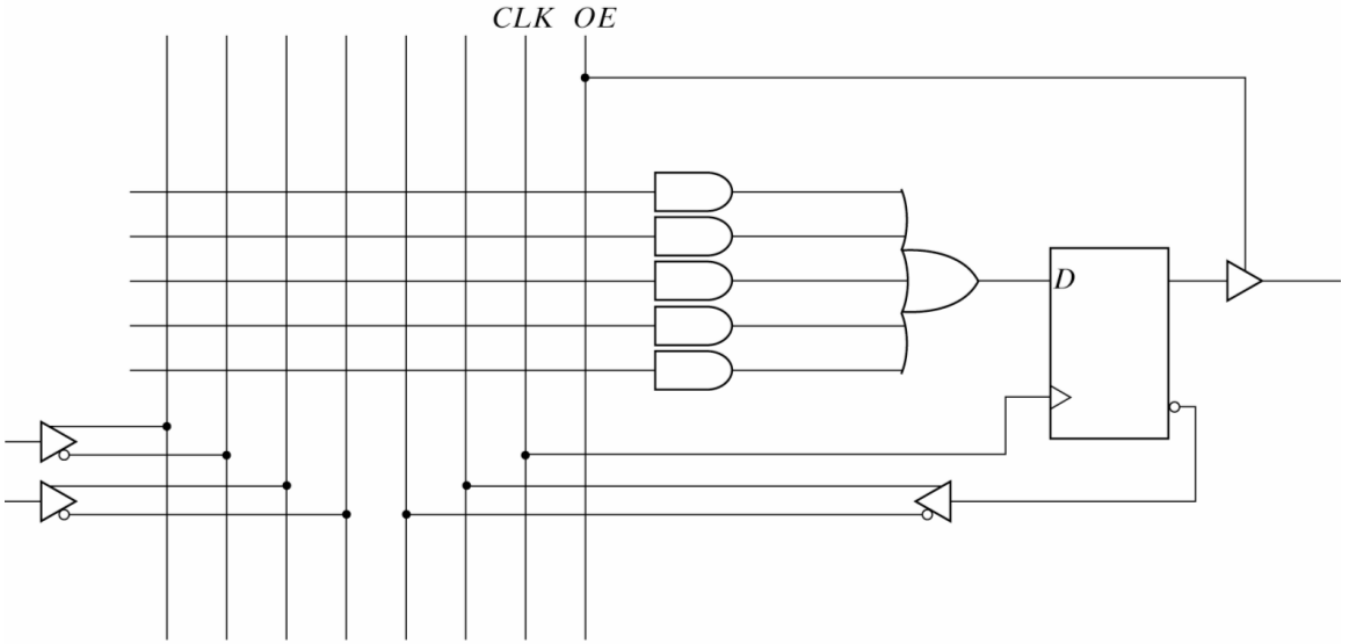
- c.f. Combinational PLDs
  - PROM, PAL, PLA
- Definition of SPLD
  - The PLD that includes both gates and flip-flops
- Type of SPLD
  - SPLD : Sequential(or Simple) programmable logic device
  - CPLD : Complex programmable logic device
  - FPLS : Field programmable logic sequencer
  - FPGA : Field programmable gate array

Block Diagram of SPLD



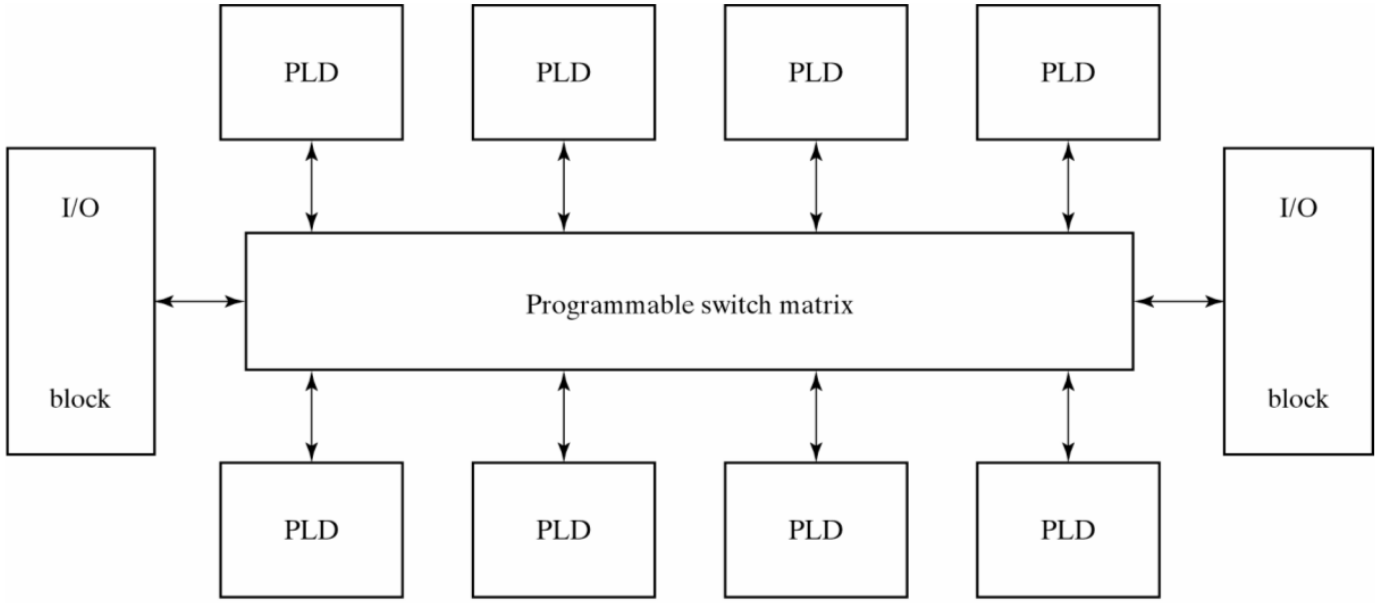
Macrocell

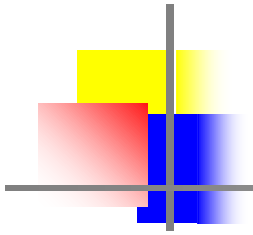
- Definition
  - Each section of an SPLD
  - Sum of product combinational logic + (optional)flip flop
- Basic logic diagram of macrocell



CPLD(Complex Programmable Logic Device)

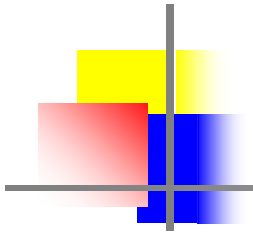
- Definition
  - A collection of individual PLDs connected through a programmable interconnection structure(e.g., switch) on a single IC
- Block diagram of CPLD

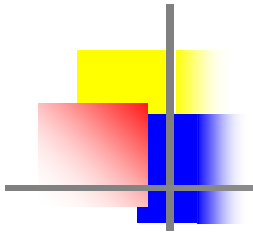


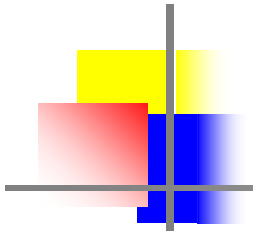


FPGA(Field Programmable Gate Array)

- Components of an FPGA
  - Logic block
    - Lookup tables -- A truth table stored in an SRAM
    - Multiplexers, Gates
    - Flip-flops
  - Programmable I/O block
  - Programmable interconnection block
- Internal truth table
  - Provides the combinational functions for the logic block
  - The functions are realized from the truth table stored in the SRAM, similar to the mannerthat combinational circuit functions are implemented with ROM







Discussion~~~