Chapter 7

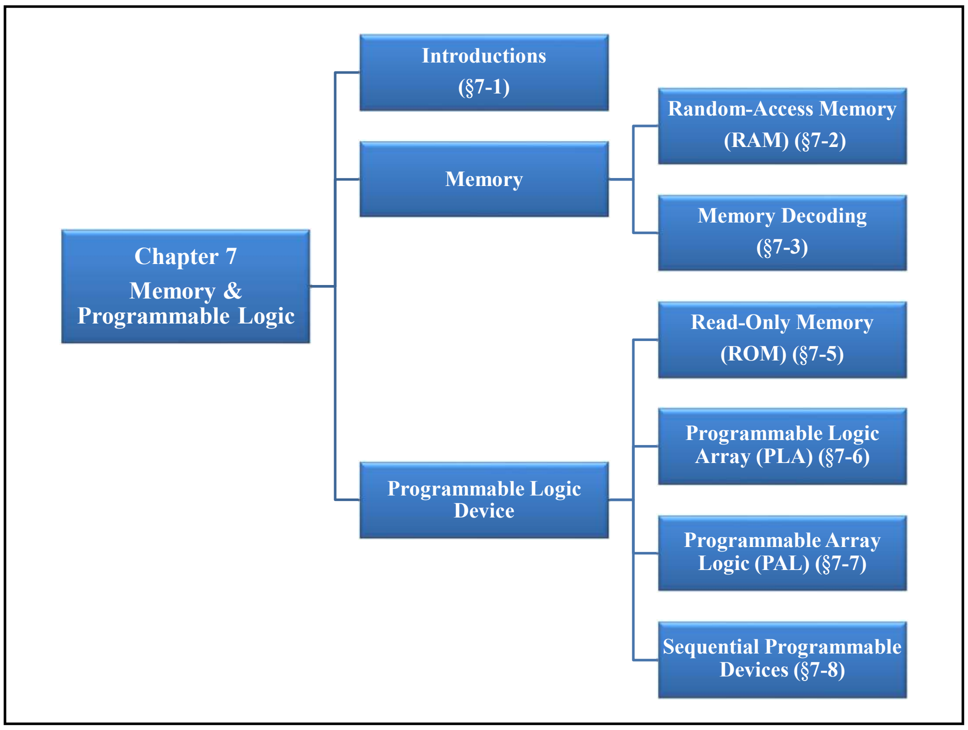
Memory & Programmable Logic

J.J. Shann

1

Contents

2


Chapter 7
Memory & Programmable Logic
Introductions (§7-1)
Memory
Random-Access Memory (RAM) (§7-2)
Memory Decoding (§7-3)
Programmable Logic Device
Read-Only Memory (ROM) (§7-5)
Programmable Logic Array (PLA) (§7-6)
Programmable Array Logic (PAL) (§7-7)
Sequential Programmable Devices (§7-8)

3

Exercises in Textbook (6th ed)

Sections	Exercises	Typical Ones
§7-2	7.1~7.4	7.1(a), 7.2(a)
§7-3	7.6~7.9, 7.15, 7.16	7.6, 7.7*, 15
§7-4	7.10~7.14	7.11
§7-5	7.17, 7.18, 7.20, 7.22	7.18(b)
§7-6	7.19, 7.21, 7.23, 7.28, 7.29	7.19
§7-7	7.24, 7.25	7.25*
§7-8	7.26, 7.27	7.26
HDL	7.5, 7.30	

* : Answers to problems appear at the end of the text.

J.J. Shann 7-4

4

7-1

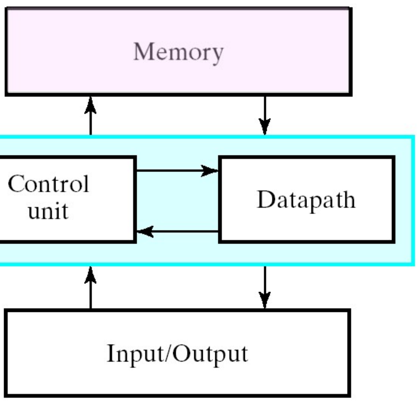
Introduction

J.J. Shann

5

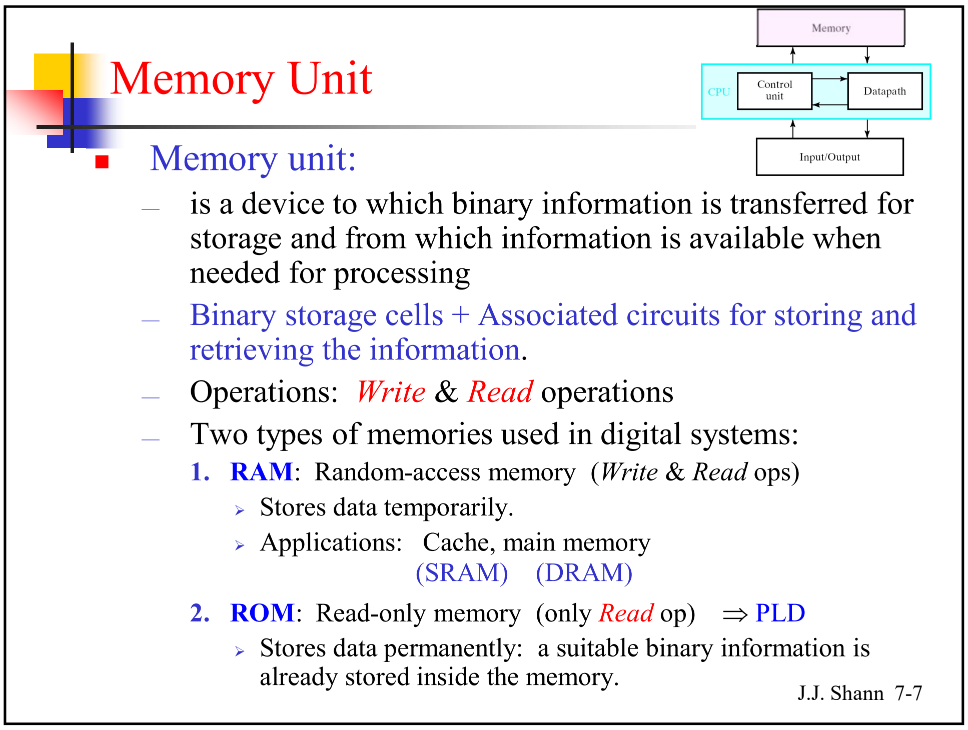
Introduction

- Digital computer:

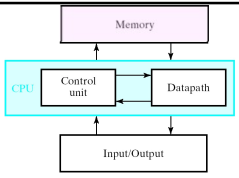


J.J. Shann 7-6

6



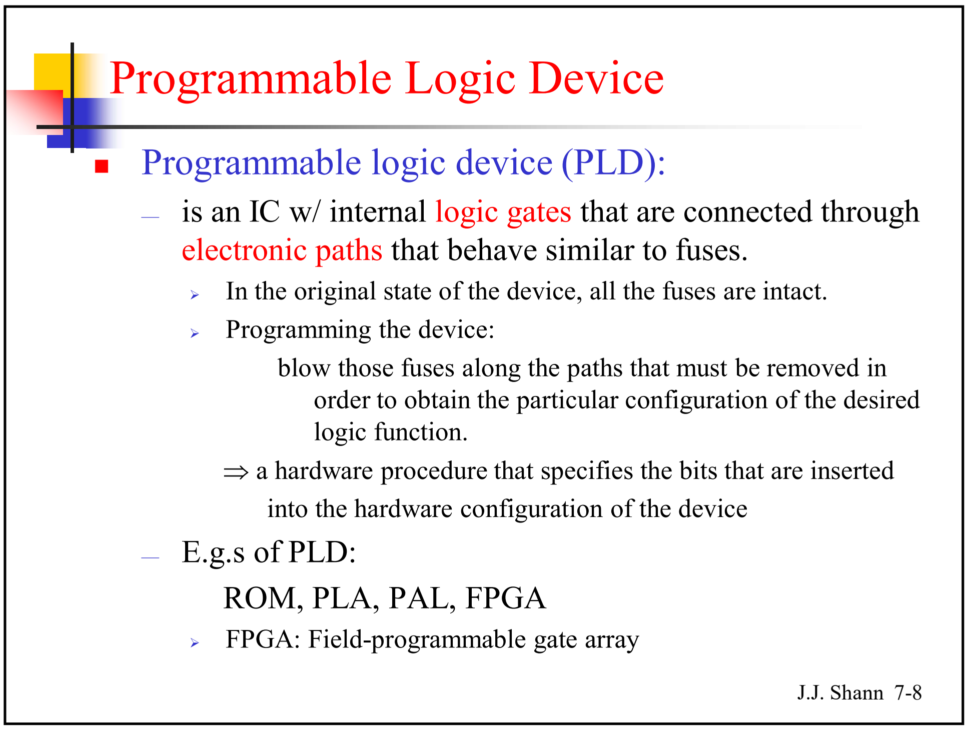
Memory Unit



- Memory unit:
  - is a device to which binary information is transferred for storage and from which information is available when needed for processing
  - Binary storage cells + Associated circuits for storing and retrieving the information.
  - Operations: *Write* & *Read* operations
  - Two types of memories used in digital systems:
    1. **RAM**: Random-access memory (*Write* & *Read* ops)
       - Stores data temporarily.
       - Applications: Cache, main memory
         (SRAM) (DRAM)
    2. **ROM**: Read-only memory (only *Read* op) ⇒ PLD
       - Stores data permanently: a suitable binary information is already stored inside the memory.

J.J. Shann 7-7

7



Programmable Logic Device

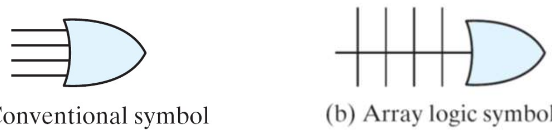
- Programmable logic device (PLD):
  - is an IC w/ internal logic gates that are connected through electronic paths that behave similar to fuses.
    - In the original state of the device, all the fuses are intact.
    - Programming the device:
      blow those fuses along the paths that must be removed in order to obtain the particular configuration of the desired logic function.

      ⇒ a hardware procedure that specifies the bits that are inserted into the hardware configuration of the device
  - E.g.s of PLD:
    ROM, PLA, PAL, FPGA
    - FPGA: Field-programmable gate array

J.J. Shann 7-8

8

- Conventional & Array logic diagrams for OR gate:



§7-5

J.J. Shann 7-9

7-2

Random-access Memory

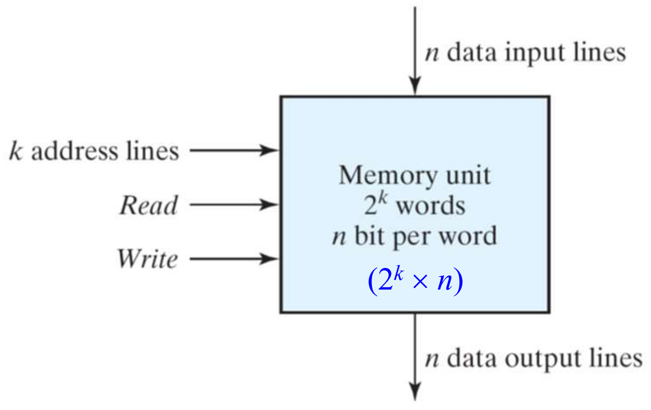
J.J. Shann

Random-access Memory

- Memory unit:
  - a collection of *storage cells* together w/ *associated ckts* needed to transfer information in and out of the device
  - *word*: an entity of bits that move in & out of storage as a unit
  - *capacity* of a memory unit: # of bytes it can store
    # words, # bits/word
- Random-access memory :
  - the time it takes to transfer information to or from any desired random location is always the same

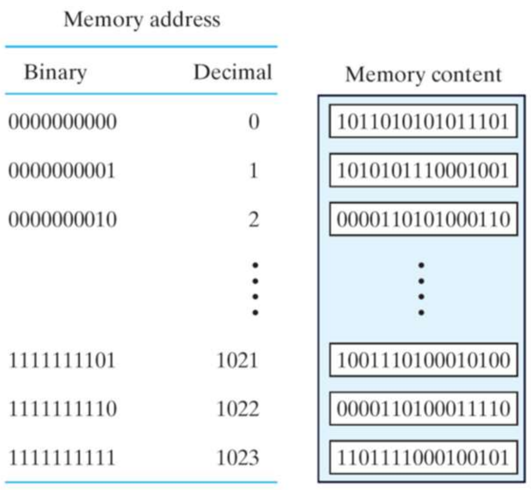
J.J. Shann 7-11

- Block diagram of a memory unit:



J.J. Shann 7-12

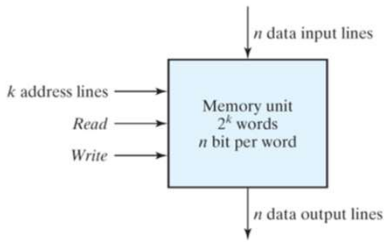
- Example: a 1024 × 16 memory ⇒ $2^{10} \times 16$



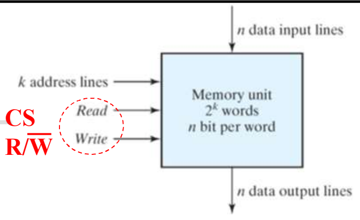
Memory address		Memory content
Binary	Decimal	
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

A. Write & Read Operations

- *Write* op: transfer-in
  - Steps:
    - Apply the binary addr of the desired word to the addr lines.
    - Apply the data bits that must be stored in memory to the data input lines.
    - Activate the *Write* input.



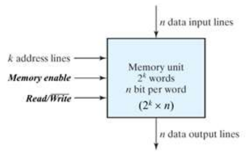
- *Read* op: transfer-out
  - Steps:
    - Apply the binary addr of the desired word to the addr lines.
    - Activate the *read* input.



- Control inputs to a RAM chip:
  - *Read* signal, *Write* signal
  - *Memory enable* (*Chip Select*), *Read/$\overline{Write}$*

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



B. Memory Description in HDL

- Memory is modeled in the Verilog HDL by an array of registers:
  - is declared with a **reg** keyword, using a ***two-dimensional array***
  - E.g.: a memory of 1024 words with 16 bits/word

    ⇒ a 2D array of 1024 registers, each containing 16 bits

```
reg [15: 0] memword [0: 1023];
```

$\underbrace{[15{:}\ 0]}_{\text{\# of bits in a word (word length)}}$ $\underbrace{[0{:}\ 1023]}_{\text{\# of words in memory (memory depth)}}$

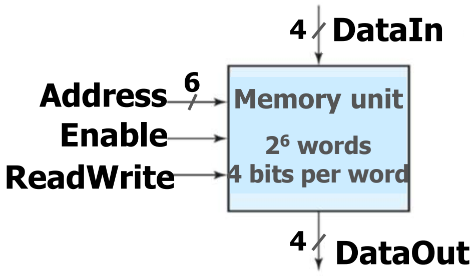
# of bits in a word (*word length*)

# of words in memory (*memory depth*)

* memword[512] refers to the 16-bit memory word at address 512.

HDL Example 7.1: Operation of a Memory Unit

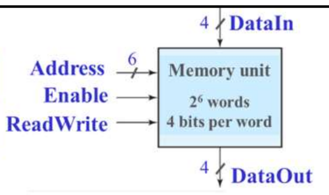
- HDL Example 7.1: Read and write operations of a memory with 64 words of 4 bits each



* The memory has three-state outputs.

Enable	ReadWrite	Memory operation
0	x	None (DataOut ← Hi-Z)
1	0	Write to selected word (Mem[Address] ← DataIn)
1	1	Read from selected word (DataOut ← Mem[Address])

17

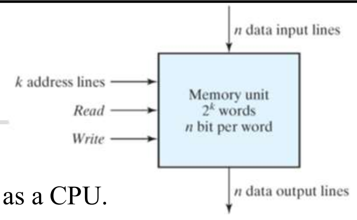


Enable	ReadWrite	Memory operation
0	x	DataOut ← Hi-Z
1	0	Mem[Address] ← DataIn
1	1	DataOut ← Mem[Address]

```
module memory (Enable, ReadWrite, Address, DataIn, DataOut);
   input           Enable, ReadWrite;
   input   [3: 0]  DataIn;
   input   [5: 0]  Address;
   output  [3:0]   DataOut;
   reg     [3: 0]  DataOut;
   reg     [3: 0]  Mem [0: 63];             //64 x 4 memory
   always @ (Enable or ReadWrite)
      if (Enable)
         if (ReadWrite) DataOut = Mem[Address];   //Read
         else Mem[Address] = DataIn;              //Write
      else DataOut = 4'bz;                  //High impedance state
endmodule
```

18

C. Timing Waveforms



- Operation of a memory unit:
  - is controlled by an external device such as a CPU.
    - The CPU is usually synchronized by its own clock.
    - The memory does not employ an internal clock.

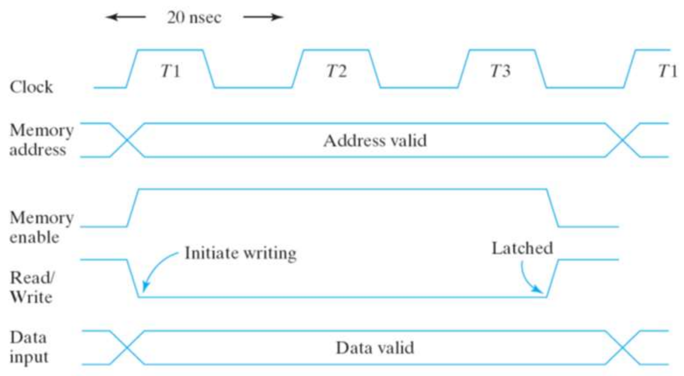
      Its read & write operations are specified by control inputs.

    ⇒ The CPU must provide the memory control signals to synchronize its internal clocked operations w/ the read and write operations of memory.
- Access time: (*)
  - *read cycle time*: the max time from the application of the addr to the appearance of the data at the Data Output
  - *write cycle time*: the max time from the application of the addr to the completion of all internal memory ops required to store a word
  - The access time of the memory must be related within the CPU to a period equal to a fixed # of CPU clock cycles.

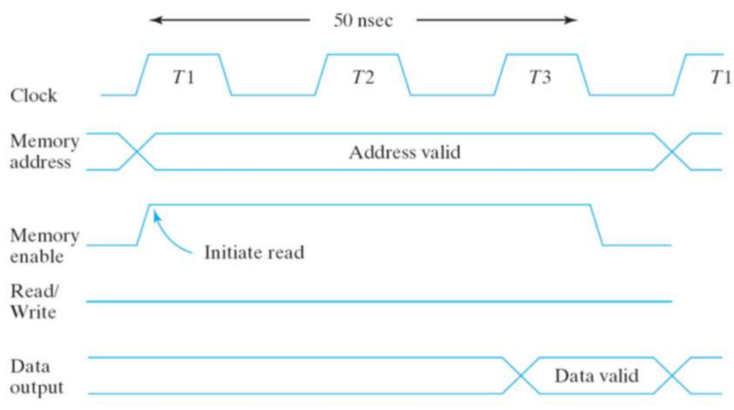
- Example: CPU – 50MHz clock frequency (20 ns)

  Memory – access time & cycle time does not exceed 50 ns

Write cycle

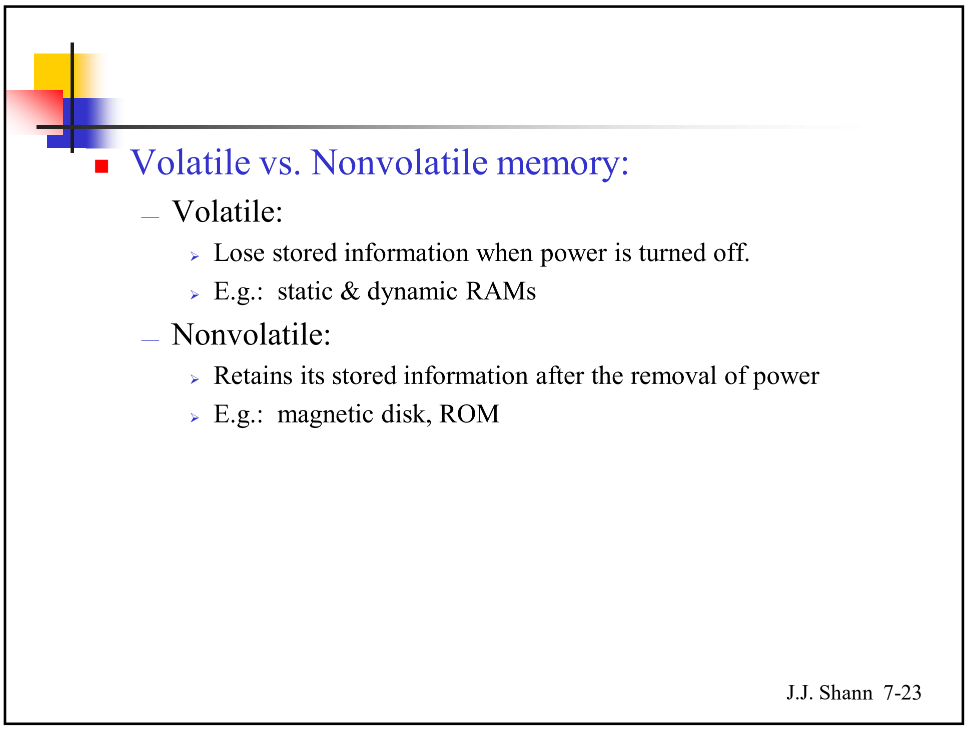


Read cycle



D. Types of Memory

- Random-access vs. Sequential-access memory:
  - Sequential-access memory: magnetic disk or tape unit
- Operating modes of RAMs:
  1. Static RAM: SRAM (Cache)
     - Consists essentially of internal latches that store the binary information.
     - The stored information remains valid as long as power is applied to the RAM.
  2. Dynamic RAM: DRAM (Main memory)
     - Stores the binary information in the form of electric charges on capacitors.
     - The stored information remains valid as long as power is applied to the RAM.

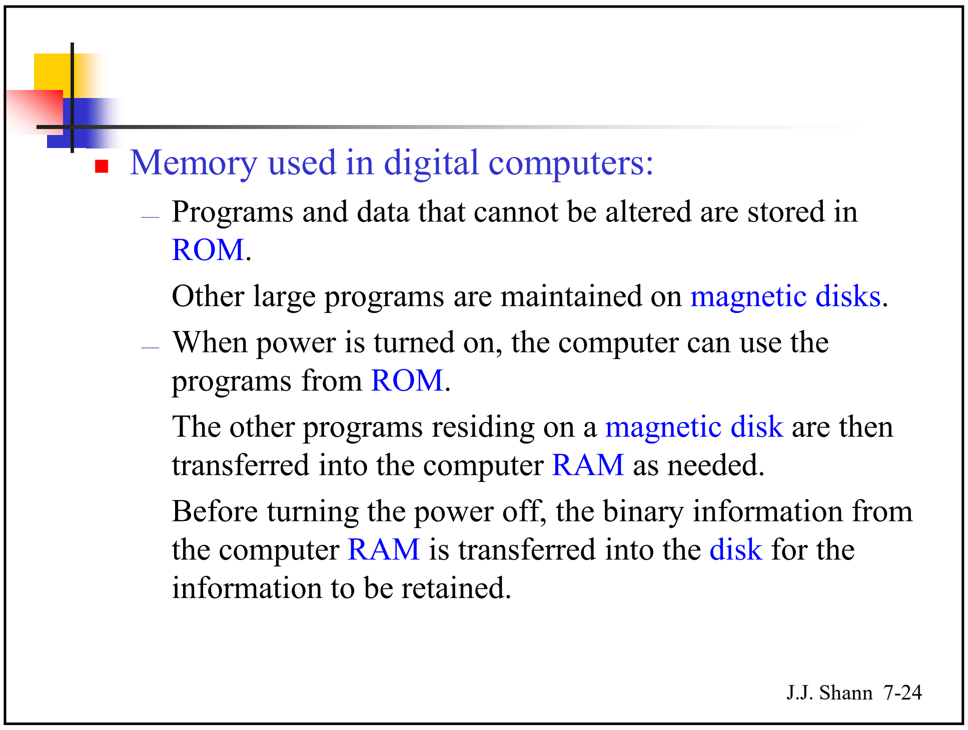


Volatile vs. Nonvolatile memory:

- Volatile:
  - Lose stored information when power is turned off.
  - E.g.: static & dynamic RAMs
- Nonvolatile:
  - Retains its stored information after the removal of power
  - E.g.: magnetic disk, ROM

J.J. Shann 7-23

23



Memory used in digital computers:

- Programs and data that cannot be altered are stored in ROM.

  Other large programs are maintained on magnetic disks.
- When power is turned on, the computer can use the programs from ROM.

  The other programs residing on a magnetic disk are then transferred into the computer RAM as needed.

  Before turning the power off, the binary information from the computer RAM is transferred into the disk for the information to be retained.

J.J. Shann 7-24

24

E. Memory Cells of RAM (§7-3 + 補充資料)

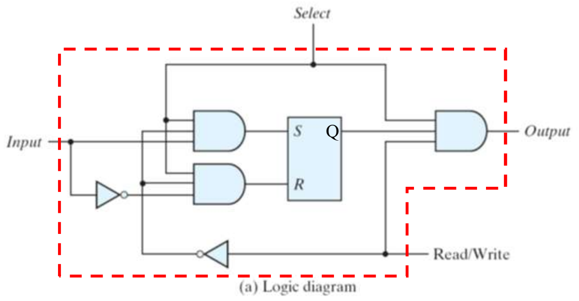
- Types of RAMs:
  1. Static RAM (SRAM): Volatile $\rightarrow$ Cache
     - Consists essentially of internal latches that store the binary information.
     - Advs.: easier to use, shorter read and write cycles, & no refreshing
  2. Dynamic RAM (DRAM): Volatile $\rightarrow$ Main memory
     - Stores the binary information in the form of electric charges on capacitors.
     - The capacitors must be periodically recharged by *refreshing* the DRAM (every few milliseconds).
     - Advs.: reduced power consumption & larger storage capacity

J.J. Shann 7-25

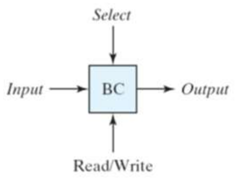
25

SRAM Cell (§7-3)

- SRAM memory cell: stores one bit of information



(a) Logic diagram



(b) Block diagram

Select	Read/Write	S	R	Q^+	Output
0	×	0	0	Q	0
1	0	Input	Input′	Input	0
1	1	0	0	Q	Q

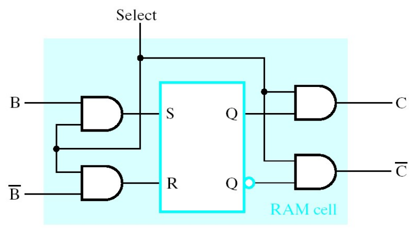
26

(Supplementary Materials)

補充資料：DRAM Cell

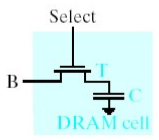
- SRAM cell vs. DRAM cell:

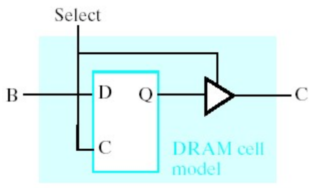
SRAM cell



– 6 transistors

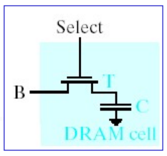
DRAM cell



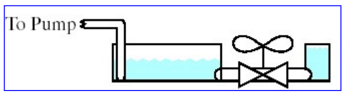


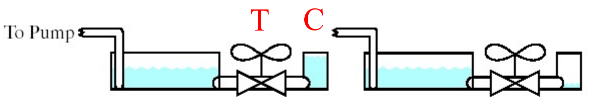
– 1 transistor (T) & 1 capacitor (C)
– Leaks ⇒ Refresh
– Destructive read ⇒ Restore

J.J. Shann 7-27



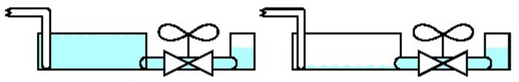
- Hydraulic analogy of DRAM cell op:





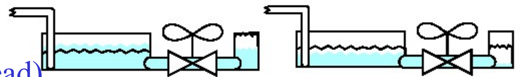
(b) (logic 1) (c) (logic 0)

Write op:



(d) a "1" is to be written (e) a "0" is to be written

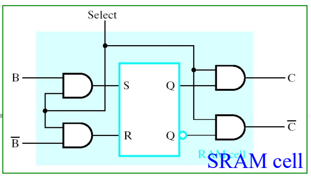
Read op:
(destructive read)

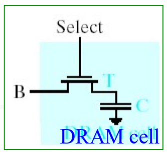


(f) the reading of "1" (g) the reading of "0"

J.J. Shann 7-28

Advs. of DRAM:





- Density: have 4 times the density of SRAM.
- Cost/bit: is 3 to 4 times less than SRAM.
- Power: lower power requirement

⇒ DRAM is the preferred technology for large memories (e.g.: main memory).

Disadv. of DRAM:

- its electronic design is considerably more challenging:
  - Destructive read ⇒ Restore
  - Leaks ⇒ Refresh

J.J. Shann 7-29

29

7-3

Memory Decoding

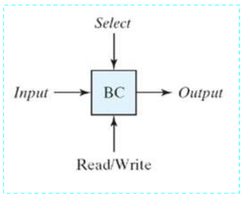
J.J. Shann

30

Memory Decoding

- Memory unit: storage components + decoding ckts
  - Decoding ckts: select the memory word specified by the input address
- RAM of m words & n bits/word:

  $m \times n$ binary storage cells + associated decoding ckts



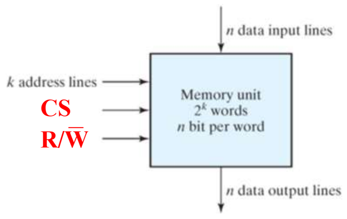
J.J. Shann 7-31

31

A. Internal Construction

- Logical construction of a RAM: 2^k words $\times$ n bits/word

  $2^k \times n$ binary storage cells + associated decoding ckts
  - Decoding ckts: *Linear decoding*
    - Requires k address lines that go into a $k \times 2^k$ decoder
    - Each one of the decoder outputs selects one word of n bits for reading or writing.

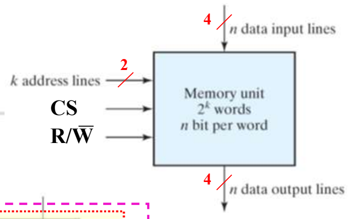


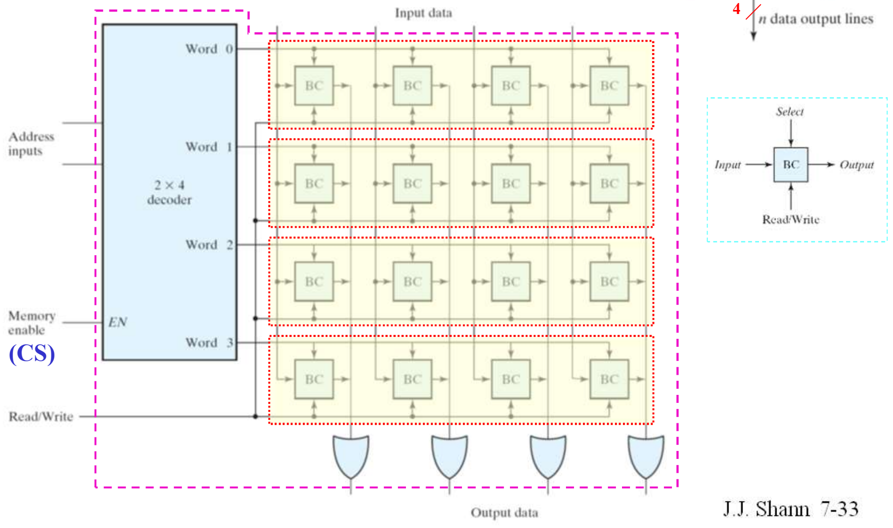
J.J. Shann 7-32

32

Example

- Example: a 4 × 4 RAM





B. Coincident decoding

- Linear decoding: 2^k words
  - Employ a $k \times 2^k$ decoder
- Coincident decoding: 2^k words
  - 2-D coincident decoding:
    - Employ two decoders in a two-dimensional selection scheme
      $\Rightarrow$ two $k/2$-input decoders (row selection & column selection)
    - * Notice: arrange the memory cells in an array that is as close as possible to **square**

	Linear	**Coincident (2D)**
Decoder	1 $k \times 2^k$	2 $k/2 \times 2^{k/2}$
# AND gates	2^k	$2 \times 2^{k/2}$ $(= 2^{k/2+1})$
# inputs/gates	k	$k/2$
Read & write times	longer	shorter

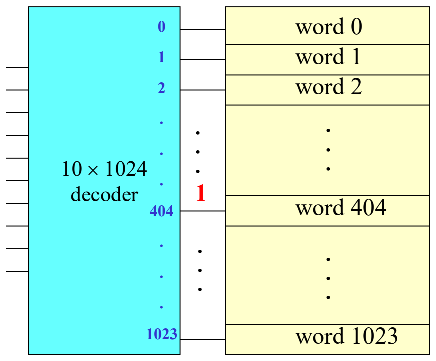
Assumption: k is an even number

Example

- Example: a 1K-word memory ⇒ 10 address lines

Linear decoding:

binary address
0110010100
(404_{10})



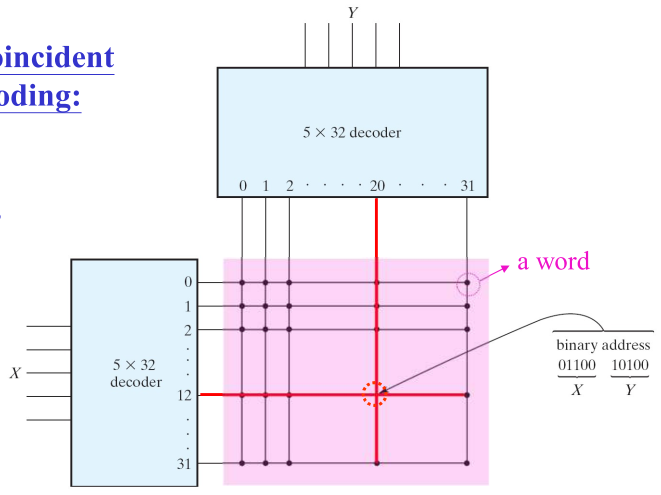
J.J. Shann 7-35

35

a 1K-word memory ⇒ 10 address lines

2-D coincident decoding:

binary address
01100 10100
X Y
(12) (20)



1024 words

J.J. Shann 7-36

36

- The savings of the coincident selection scheme:
  - A single 10-to-1024 decoder:
    1024 AND gates w/ 10 inputs in each
  - Coincident selection: two 5-to-32 decoders
    - 2 × (32 AND gates w/ 5 inputs in each)
    - ⇒ 64 AND gates w/ 5 inputs in each

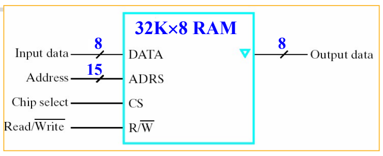
	Linear	**Coincident (2D)**
Decoder	1 $k \times 2^k$	2 $k/2 \times 2^{k/2}$
# AND gates	2^k	$2 \times 2^{k/2}$ ($= 2^{k/2+1}$)
# inputs/gates	k	$k/2$
Read & write times	longer	shorter

J.J. Shann 7-37

37

Example

* Arrange the memory cells in an array that is as close as possible to **square.**



- E.g.: For a 32K×8 RAM
  - Linear selection scheme:
    - A single 15-to-2^{15} line decoder:
      ⇒ 32,768 AND gates w/ 15 inputs in each
  - Coincident selection:
    - Make the # of rows and columns in the array equal:
      (*Arrange the memory cells in the array as close to square as possible.)
      32K × 8 = 256K bits = 2^{18} bits = $2^9 \times 2^9$

$$= \underbrace{2^9}_{\text{row selection}} \times \underbrace{(2^6 \times 8)}_{\text{column selection}} \text{ bits}$$

⇒ a 9-to-512 line decoder (row) & a 6-to-64 line decoder (column)
⇒ 512 9-input AND gates & 64 6-input AND gates
⇒ 576 AND gates (with 9- or 6-input in each)

J.J. Shann 7-38

38

C. Address Multiplexing

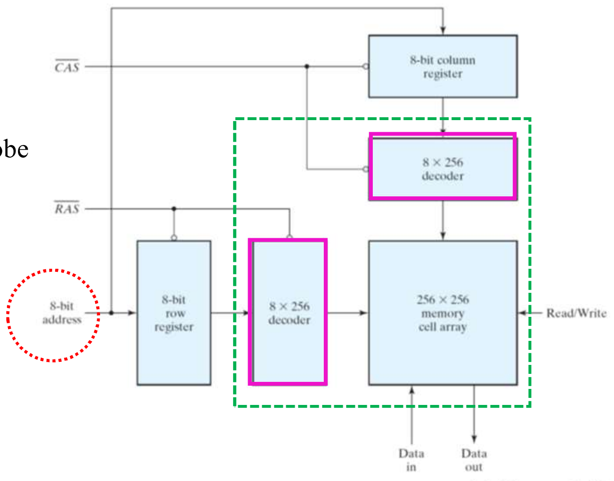
- Address multiplexing:
  - Reduce the # of pins in the IC package
  - Use one set of address input pins accommodates the address components.
  - E.g.: In a 2-dimensional array, the same set of pins is used for both of the row and column addresses

Example

- Example: a 64K DRAM (64K = 2^{16})

$\overline{\text{RAS}}$: row addr strobe

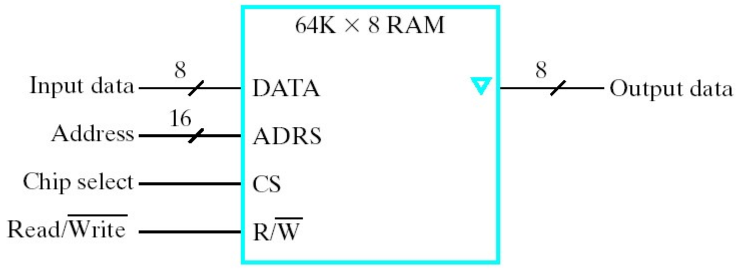
$\overline{\text{CAS}}$: column addr strobe



(Supplementary Materials)

補充資料：Array of SRAM ICs

- Memory unit:
  - Capacity: # words, # bits/word
- Symbol for a RAM chip:
  - E.g.: 64K × 8 RAM



J.J. Shann 7-41

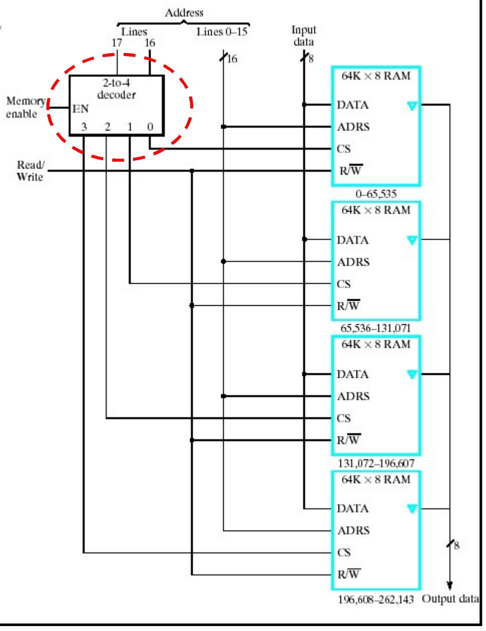
41

Increasing # of Words

- E.g.:

  Construct a 256K×8 RAM by using 64K×8 RAM ICs

  (2^{18})
  256K×8 RAM
  ⇓
  four 64K×8 RAM ICs
  (2^{16})



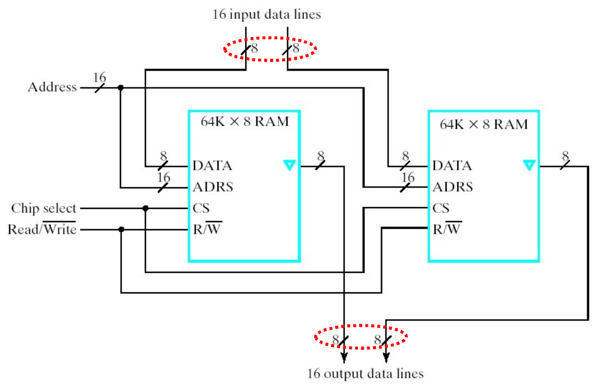
42

Increasing # Bits/Word in the Memory

- E.g.:

  Construct a 64K×16 RAM by using 64K×8 RAM ICs

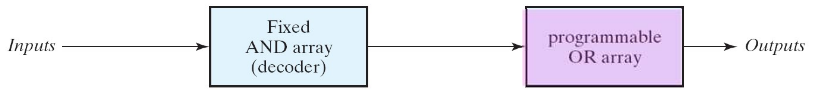
  64K×16 RAM ⇒ two 64K×8 RAM ICs



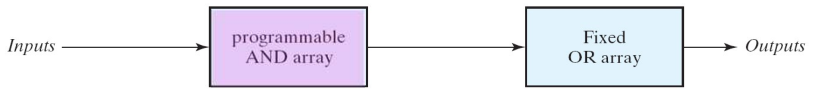
* Increasing both # words & # bits/word?

7-5 ~ 7-7 Combinational PLDs

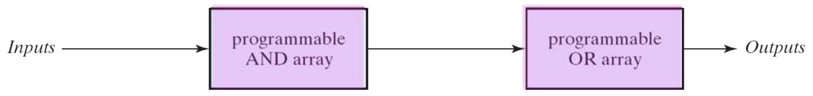
- Combinational PLD:
  - is an IC w/ *programmable gates* divided into an AND array and an OR array to provide an AND-OR sum of product implementation.
- 3 major types of combinational PLDs:
  - Differ in the placement of the programmable connections in the AND-OR array.
  1. PROM: programmable read-only memory (§7-5)
  2. PAL: programmable array logic (§7-7)
  3. PLA: programmable logic array (§7-6)



(a) Programmable read-only memory (PROM)



(b) Programmable array logic (PAL)



(c) Programmable logic array (PLA)

J.J. Shann 7-57

57

7-5

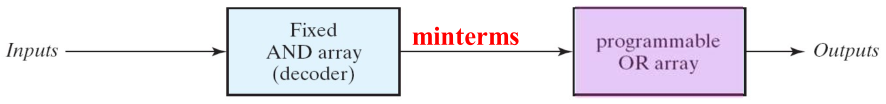
Read-Only Memory (ROM)

J.J. Shann

58

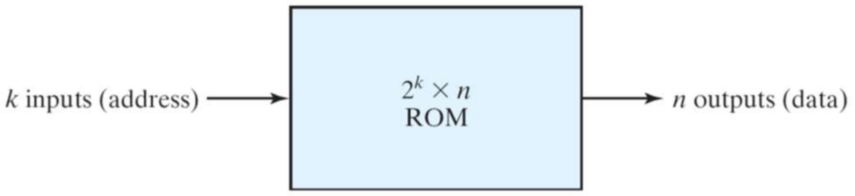
Read-Only Memory (ROM)

- ROM:
  - A memory device in which permanent binary information is stored



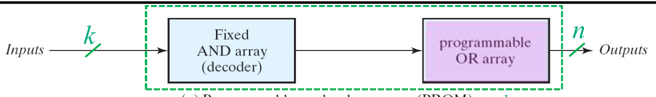
(a) Programmable read-only memory (PROM)

- ROM Block diagram:



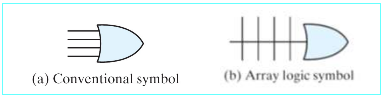
2^k words, n bits/word

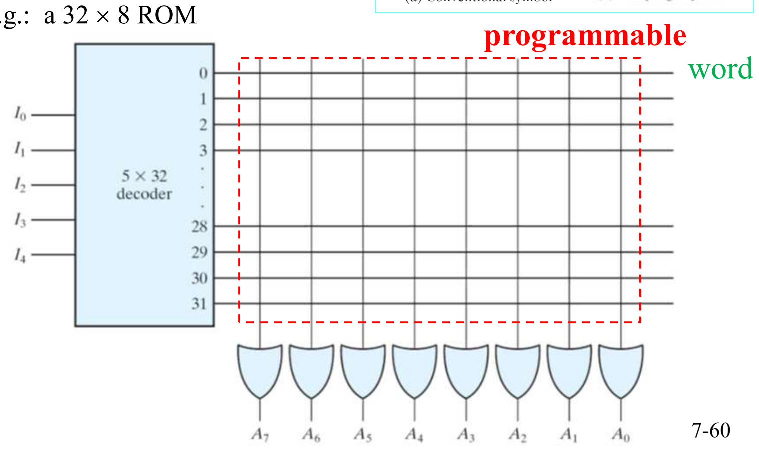
J.J. Shann 7-59



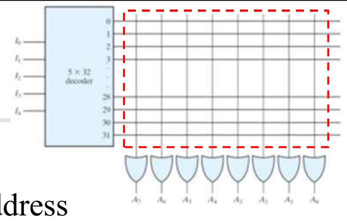
(a) Programmable read-only memory (PROM) $2^k \times n$ ROM

- Internal logic of a $2^k \times\ n$ ROM: comb. ckt.
  - have an internal $k \times 2^k$ decoder & n OR gates
  - E.g.: a 32 × 8 ROM





7-60



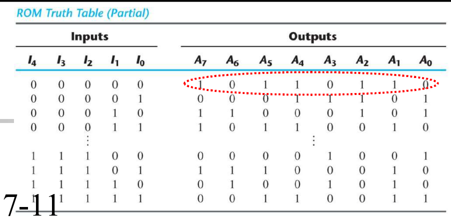
ROM truth table:

- shows the word content in each address
- gives all the information for programming the ROM
- E.g.:

ROM Truth Table (Partial)

Inputs					Outputs							
I_4	I_3	I_2	I_1	I_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
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

61

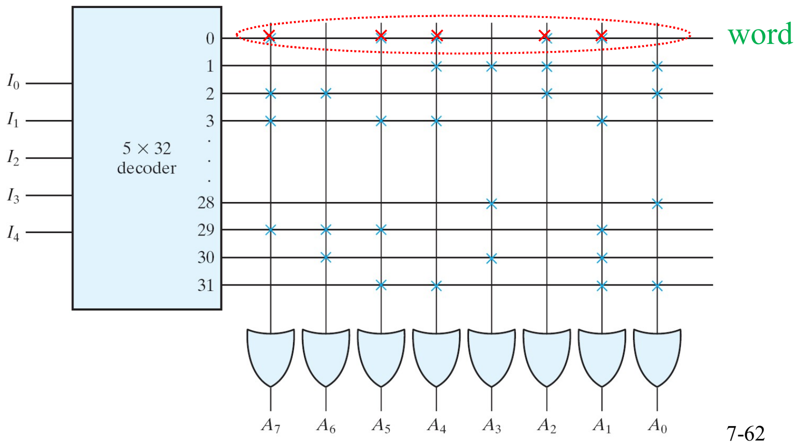


ROM Truth Table (Partial)

Inputs					Outputs							
I_4	I_3	I_2	I_1	I_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
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

Programming the ROM:

- E.g.: Table 7-3 $\Rightarrow$ Figure 7-11



7-62

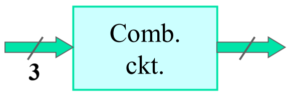
Combinational Circuit Implementation

- ROM: a decoder + OR gates
  - Boolean functions $\rightarrow$ "sum of minterms" form
    $\rightarrow$ ROM truth table
    $\rightarrow$ ROM implementation
  - For an n-input, m-output combinational ckt
    $\Rightarrow 2^n \times m$ ROM
- Design procedure:
  1. Determine the size of ROM required from the # of inputs and outputs of the comb. ckt.
  2. Obtain the programming truth table of the ROM.
  3. The 0's (or 1's) in the output functions of the truth table directly specify those links that must be removed to provide the required comb ckt in sum of minterms form.

J.J. Shann 7-63

63

Example 7-1



- E.g.: Design a combinational ckt using a ROM.
  The ckt accepts a 3-bit number and generates an output binary number equal to the square of the input number.

<Ans.>

Truth table:

Inputs			Outputs						
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	Decimal
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

3 inputs & 6 outputs
$\Rightarrow 2^3 \times 6$ ROM
$\Rightarrow$ 8 words, 6 bits/word

$\boxed{B_1 = 0}$ $\boxed{B_0 = A_0}$

J.J. Shann 7-64

64

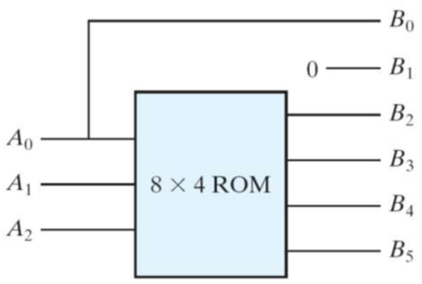
Inputs			Outputs						
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

ROM implementation:

$B_1 = 0$, $B_0 = A_0$

Truth table (Table 7-4) $\Rightarrow$ 3 inputs, 4 outputs

$\Rightarrow$ a 8×4 ROM



(a) Block diagram

A_2	A_1	A_0	B_5	B_4	B_3	B_2
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

(b) ROM truth table

Types of ROMs

3

- Types of ROM:
  - Mask programming
  - PROM: programmable ROM
  - EPROM: erasable PROM
  - EEPROM: electrically-erasable PROM

7-6

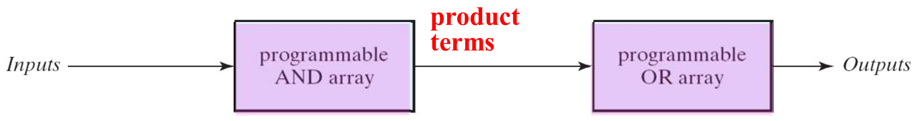
Programmable Logic Array

J.J. Shann

67

Programmable Logic Array

- PLA:



(c) Programmable logic array (PLA)

- The array of AND gates can be programmed to generate any product terms (AND terms) of the input variables.
- The product terms are then connected to OR gates to provide the sum of products for the required Boolean functions.

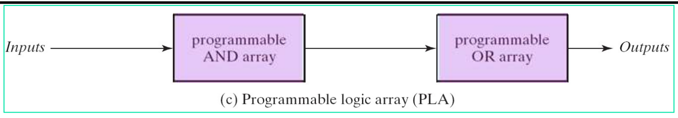
* Compare to ROM:

does not provide full decoding of the variables

⇒ does not generate all the minterms

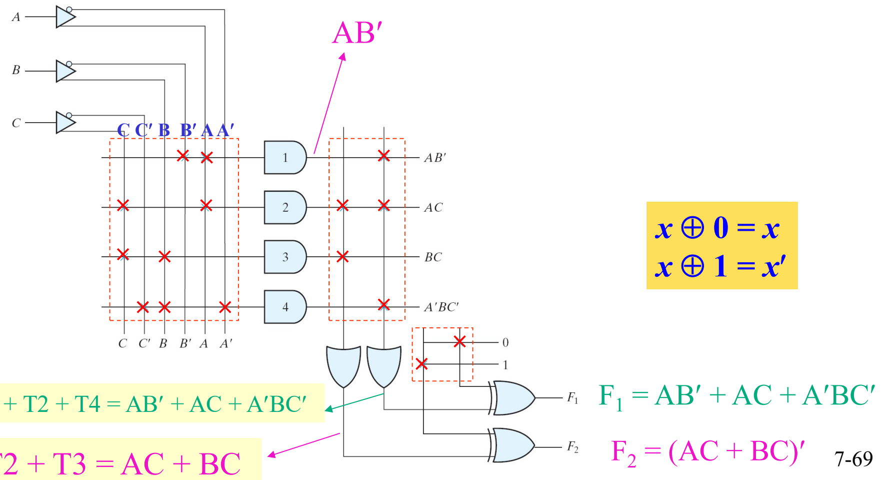
J.J. Shann 7-68

68



(c) Programmable logic array (PLA)

- Internal logic of a PLA: *(# inputs × # product terms × # outputs)*
  - E.g.: a 3 × 4 × 2 PLA



$x \oplus 0 = x$

$x \oplus 1 = x'$

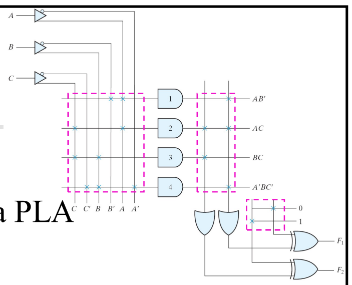
$F_1 = AB' + AC + A'BC'$

$F_2 = (AC + BC)'$

7-69

69

- PLA programming table:
  - shows the internal connections of a PLA
  - E.g.: Figure 7-14 ⇒ Table 7-5



PLA Programming Table

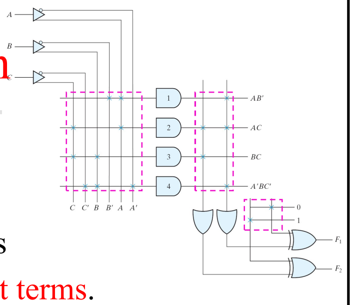
		Inputs			Outputs (T)	Outputs (C)
	Product Term	*A*	*B*	*C*	*F*$_1$	*F*$_2$
AB′	1	1	0	—	1	—
AC	2	1	—	1	1	1
BC	3	—	1	1	—	1
A′BC′	4	0	1	0	1	—

(AND array) (OR array)

J.J. Shann 7-70

70

Comb Ckt Implementation



- Design method:
  - **Sum-of-products form**
  - A PLA has a finite # of AND gates

    ⇒ Reduce the # of ***distinct*** product terms.
  - Both the true and complement of each function should be simplified to see which one can be expressed w/ fewer product terms and which one provides product terms that are common to other functions.

  (* The # of literals in a term is not important.)

J.J. Shann 7-71

71

Example 7-2

- E.g.: Implement the following two Boolean functions w/ a PLA:

  $F_1(A, B, C) = \Sigma\,(0, 1, 2, 4)$

  $F_2(A, B, C) = \Sigma\,(0, 5, 6, 7)$

<Ans.>

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

$F_1 = \overline{A}\,\overline{B} + \overline{A}\,\overline{C} + \overline{B}\,\overline{C}$

$\overline{F_1} = AB + AC + BC$

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

$F_2 = AB + AC + \overline{A}\,\overline{B}\,\overline{C}$

$\overline{F_2} = \overline{A}C + \overline{A}B + A\overline{B}\,\overline{C}$

72

$F_1 = \overline{A}\,\overline{B} + \overline{A}\,\overline{C} + \overline{B}\,\overline{C}$

$\overline{F_1} = AB + AC + BC$

$F_2 = AB + AC + \overline{A}\,\overline{B}\,\overline{C}$

$\overline{F_2} = \overline{A}C + \overline{A}B + A\overline{B}\,\overline{C}$

$F_1 = A'B' + A'C' + B'C'$ (6) (4) (6) $F_2 = AB + AC + A'B'C'$

$F_1 = (AB + AC + BC)'$ (6) $F_2 = (A'C + A'B + AB'C')'$

PLA programming table

	Product term	Inputs A	B	C	Outputs (C) F_1	(T) F_2
AB	1	1	1	–	1	1
AC	2	1	–	1	1	1
BC	3	–	1	1	1	–
$\overline{A}\,\overline{B}\,\overline{C}$	4	0	0	0	–	1

J.J. Shann 7-73

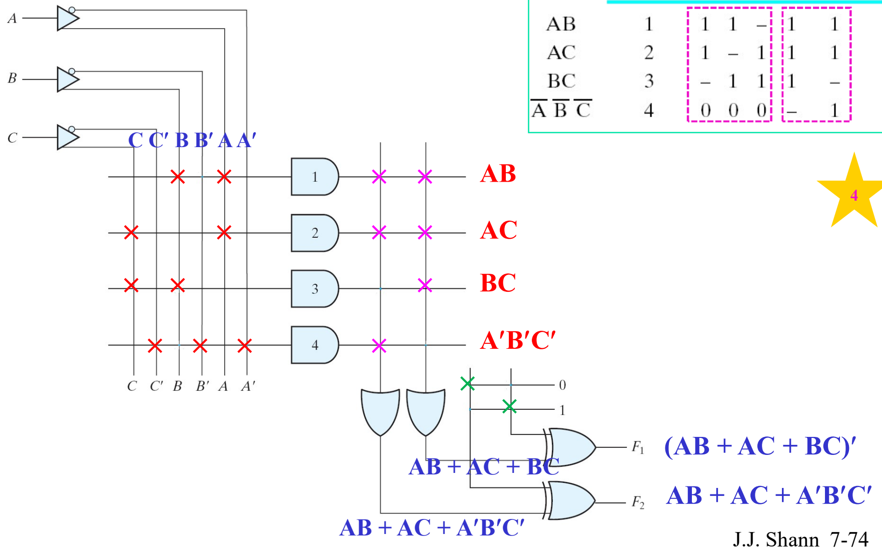
73

$F_1 = (AB + AC + BC)'$

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

PLA programming table

	Product term	Inputs A	B	C	Outputs (C) F_1	(T) F_2
AB	1	1	1	–	1	1
AC	2	1	–	1	1	1
BC	3	–	1	1	1	–
$\overline{A}\,\overline{B}\,\overline{C}$	4	0	0	0	–	1



4

J.J. Shann 7-74

74

7-7

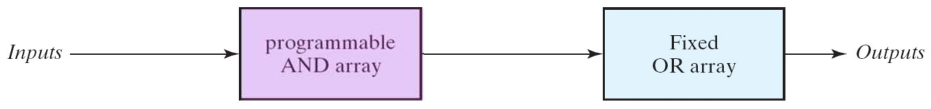
Programmable Array Logic

J.J. Shann

75

Programmable Array Logic

- PAL:

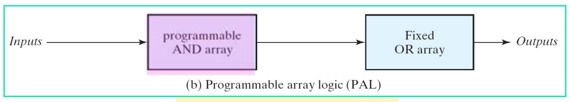


(b) Programmable array logic (PAL)

  - Only the AND gates are programmable.
  - The PAL is easier to program, but is not as flexible as the PLA.

J.J. Shann 7-76

76



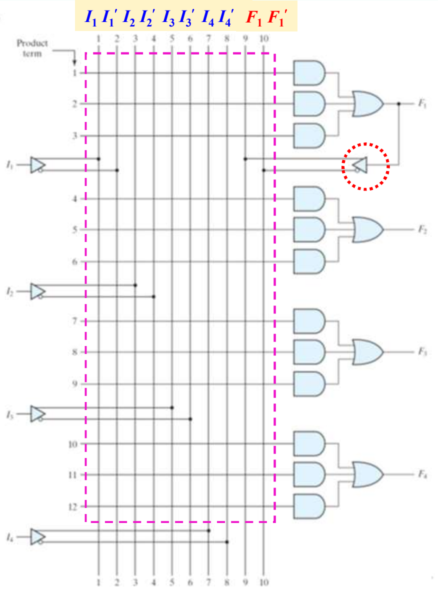
(b) Programmable array logic (PAL)

- Internal logic of a PAL:
  - E.g.:

    4 inputs

    4 outputs (4 sections)

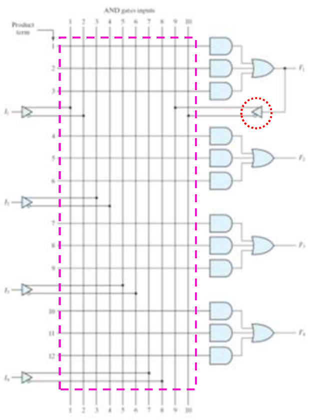
    3-wide AND-OR structure



Combination Circuit Implementation

- Design method:
  - The Boolean functions must be simplified to fit into each section.
  - Each function can be simplified by itself.

    (No sharing of product terms)
  - If the # of terms in a function is too large, it may be necessary to use two sections to implement the function.
- PAL programming table



Example

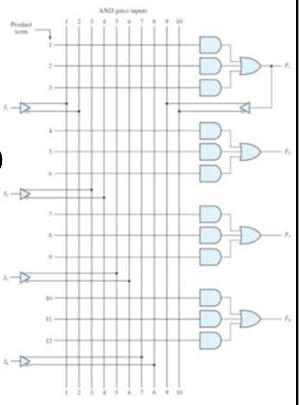
- Implement the following Boolean functions w/ a PAL:

$w(A,B,C,D) = \Sigma(2,12,13)$

$x(A,B,C,D) = \Sigma(7,8,9,10,11,12,13,14)$

$y(A,B,C,D) = \Sigma(0,2,3,4,5,6,7,8,10,11,15)$

$z(A,B,C,D) = \Sigma(1,2,8,12,13)$



<Ans.>

$w = ABC' + A'B'CD'$

$x = A + BCD$

$y = A'B + CD + B'D'$

$z = \underline{ABC' + A'B'CD'} + AC'D' + A'B'C'D$

$= w + AC'D' + A'B'C'D$

$W = ABC' + A'B'CD'$

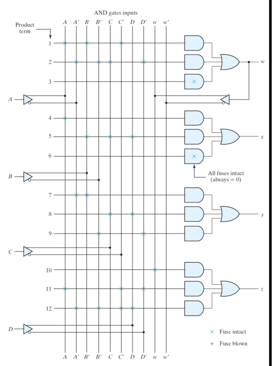
$X = A + BCD$

$Y = A'B + CD + B'D'$

$Z = W + AC'D' + A'B'C'D$

PAL programming table

Product term	AND Inputs A	B	C	D	W	Outputs
1	1	1	0	—	—	$W = AB\overline{C}$
2	0	0	1	0	—	$+\overline{A}\,\overline{B}C\overline{D}$
3	—	—	—	—	—	
4	1	—	—	—	—	$X = A$
5	—	1	1	1	—	$+BCD$
6	—	—	—	—	—	
7	0	1	—	—	—	$Y = \overline{A}B$
8	—	—	1	1	—	$+CD$
9	—	0	—	0	—	$+\overline{B}\,\overline{D}$
10	—	—	—	—	1	$Z = W$
11	1	—	0	0	—	$+A\overline{C}\,\overline{D}$
12	0	0	0	1	—	$+\overline{A}\,\overline{B}\,\overline{C}D$



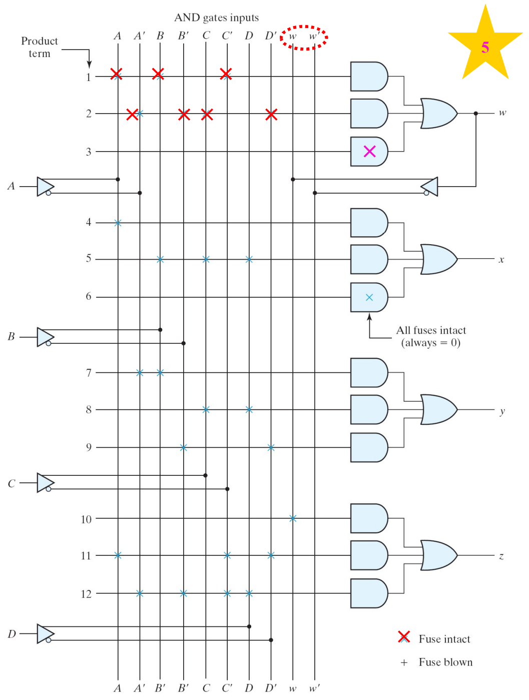
$W = ABC' + A'B'CD'$

$X = A + BCD$

$Y = A'B + CD + B'D'$

$Z = W + AC'D' + A'B'C'D$

Product term	A	B	C	D	W	Outputs
1	1	1	0	—	—	$W = AB\overline{C}$
2	0	0	1	0	—	$+\overline{A}\,\overline{B}C\overline{D}$
3	—	—	—	—	—	
4	1	—	—	—	—	$X = A$
5	—	1	1	1	—	$+BCD$
6	—	—	—	—	—	
7	0	1	—	—	—	$Y = \overline{A}B$
8	—	—	1	1	—	$+CD$
9	—	0	—	0	—	$+\overline{B}\,\overline{D}$
10	—	—	—	—	1	$Z = W$
11	1	—	0	0	—	$+A\overline{C}\overline{D}$
12	0	0	0	1	—	$+\overline{A}\,\overline{B}\,\overline{C}D$



7-8

Sequential Programmable Devices

J.J. Shann

Sequential Programmable Devices

- Combinational PLD: (§7-5 ~ 7-7)
  - Consists of only gates
- Sequential PLD:
  - Include both gates & flip-flops
  - The device can be programmed to perform a variety of sequential-ckt functions
- Three major types of sequential PLD:
  - SPLD: Sequential (or simple) PLD
  - CPLD: Complex PLD
  - FPGA: Field programmable gate array

  $\Rightarrow$ Require extensive CAD tools for their synthesis procedure.

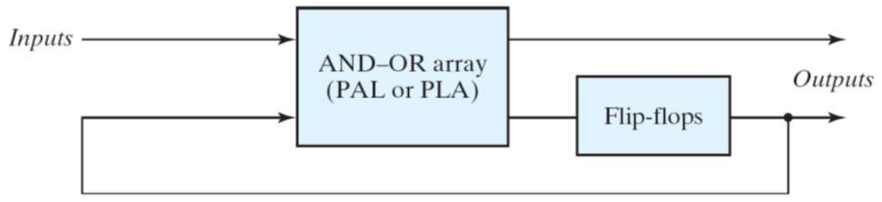
J.J. Shann 7-83

83

A. SPLD

- Sequential (simple) programmable logic device:
  - a PAL (or PLA) + a number of flip-flops

(Comb. ckt.)

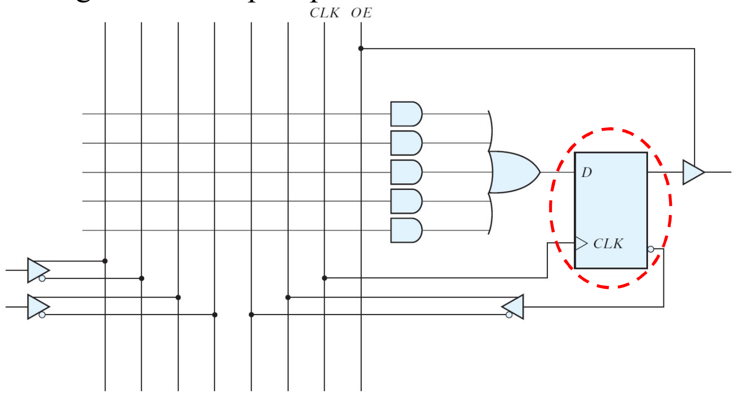


J.J. Shann 7-84

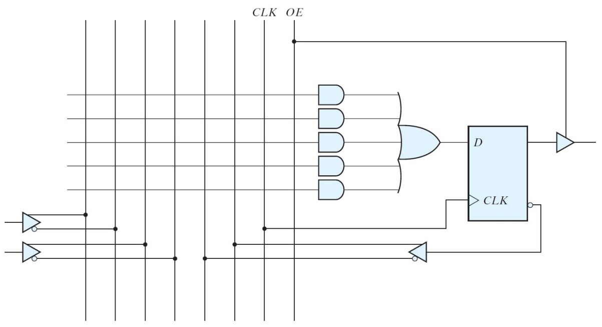
84

Macrocell: a section of an SPLD

- Contains any one of the two-level comb logic function & an optional flip-flop
- E.g.: PAL + flip-flops

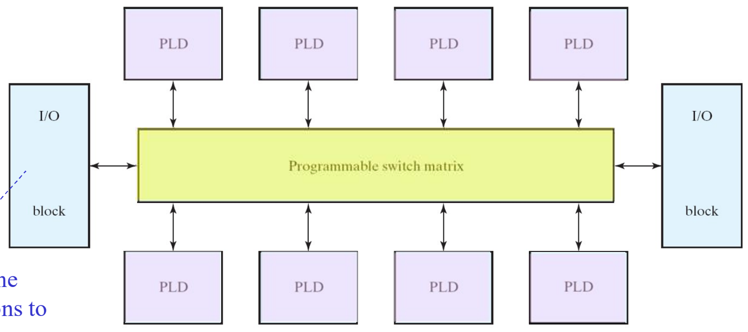


- Programming features:
  - AND array
  - ability to either use or bypass the flip-flop
  - selection of clock edge polarity
  - selection of preset and clear for the register
  - selection of the true or complement of an output



B. CPLD

- Complex PLD:
  - A collection of individual PLDs on a single integrated ckt
  - Has a programmable interconnection structure that allows the PLDs to be connected to each other.
- General CPLD configuration:



Provide the connections to the IC pins.

J.J. Shann 7-87

87

C. FPGA

- Field programmable gate array:
  - is a VLSI ckt that can be programmed in the user's location.
  - consists of an array of hundreds or thousands of logic blocks, surrounded by programmable input and output blocks and connected together via programmable interconnections.
    - An FPGA logic block:

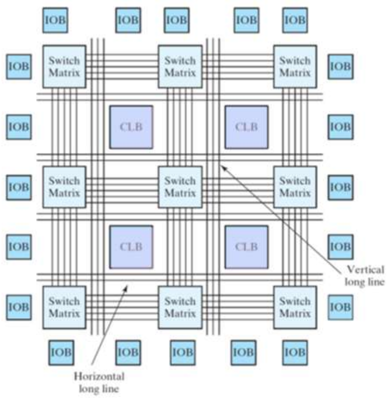
      consists of look-up tables (LUTs), multiplexers, gates, and flip-flops.

J.J. Shann 7-88

88

Xilinx FPGA

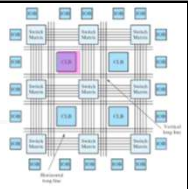
- Basic Xilinx Architecture:
  - consists of
    - an array of configurable logic blocks (CLBs)
    - a variety of local and global routing resources
    - input-output blocks (IOBs)
    - programmable I/O buffers
    - a SRAM-based configuration memory
  - E.g.: Xilinx Spartan



J.J. Shann 7-89

89

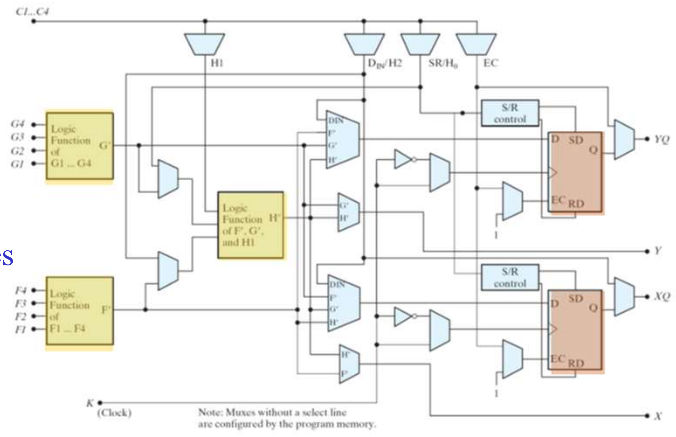
Configurable logic block (CLB)



- CLB:
  - consists of a programmable lookup table, multiplexers, registers, and paths for control signals

3 LUTs:
F: 4-input LUT
G: 4-input LUT
H: 3-input LUT

2 storage devices

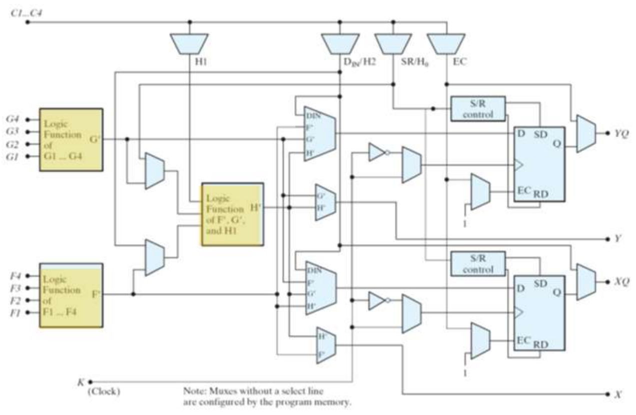

C1...C4
H1
D_{IN}/H2
SR/H_0
EC
G4
G3
G2
G1
Logic Function of G1 ... G4
G'
Logic Function of F', G', and H1
H'
F4
F3
F2
F1
Logic Function of F1 ... F4
F'
DIN
F'
G'
H'
S/R control
D
SD
Q
EC
RD
YQ
Y
XQ
X
K (Clock)
Note: Muxes without a select line are configured by the program memory.

90

90

Distributed RAM

- Distributed RAM:
  - The three function generators within a CLB can be used as either a 16×2 dual-port RAM or a 32 × 1 single-port RAM.

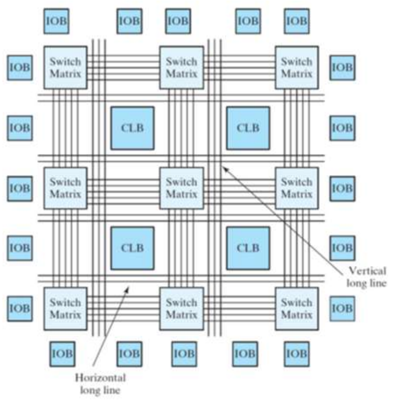

C1...C4
H1
D_{IN}/H2
SR/H_0
EC
S/R control
G4
G3
G2
G1
Logic Function of G1 ... G4
G'
Logic Function of F', G', and H1
H'
F4
F3
F2
F1
Logic Function of F1 ... F4
F'
DIN
D
SD
Q
EC
RD
YQ
Y
XQ
X
K (Clock)
Note: Muxes without a select line are configured by the program memory.

91

91

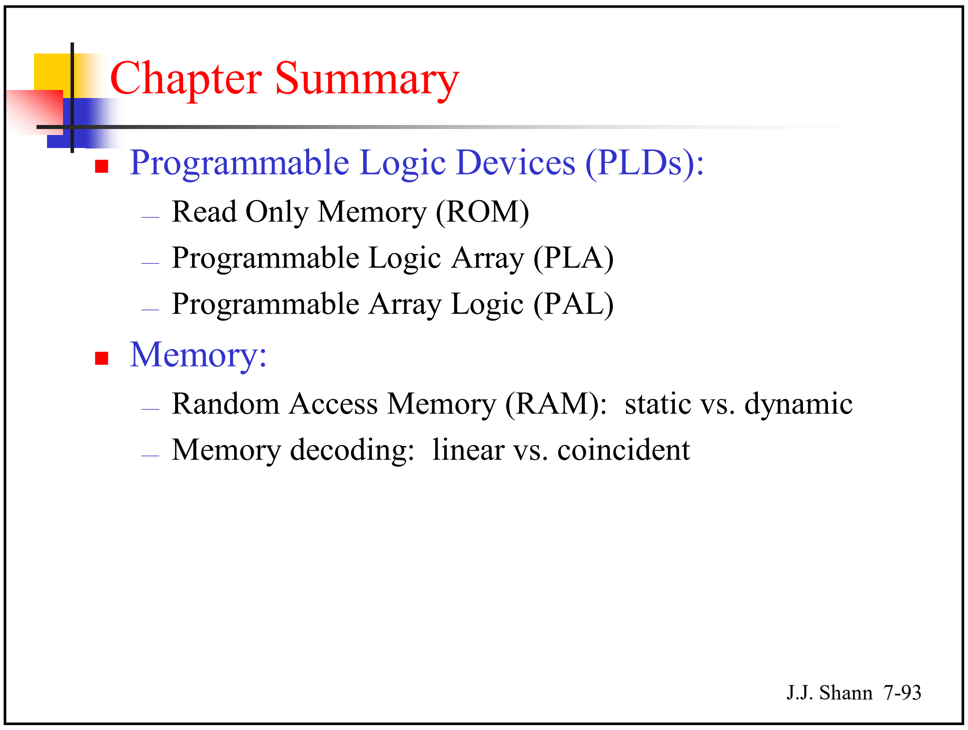
Interconnect Resources

- Interconnect resources:
  - a grid of switch matrices
  - 3 types of general-purpose interconnects:
    - single-length lines
    - double-length lines
    - long lines



J.J. Shann 7-92

92



Chapter Summary

- Programmable Logic Devices (PLDs):
  - Read Only Memory (ROM)
  - Programmable Logic Array (PLA)
  - Programmable Array Logic (PAL)
- Memory:
  - Random Access Memory (RAM): static vs. dynamic
  - Memory decoding: linear vs. coincident