جامعة بيرزيت

BIRZEIT UNIVERSITY

ANSWER BOOKLET

Student: Digital Number 9

Course: Department: Number:

Division: Instructor:

Date: Chapter 7 (PLDs)

Day Month Year

For Instructor's Use

Question	Grade
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
Total	

Chapter 7 : Memory and Programmable Logic

- memory unit : it is a device to which binary information is stored.

- Two types of memories :

1. RAM (Random access memory) :
   - Same access time for accessing any location (opposite to sequential access).
   - Read and write
   - information will be lost when power down.
   - many types (e.g SRAM, DRAM) ← volatile memory

2. ROM (Read Only Memory)
   - Read only (no write operation)
   - nonvolatile memory
   - many types (EPROM, PROM, EEPROM, Flash).
   - ROM is a programmable logic device (PLD).

⊛ RAM

- random access memory (vs sequential access memory)
  same access time.
- Read & write operations
- volatile memory
- many types (SRAM, DRAM)

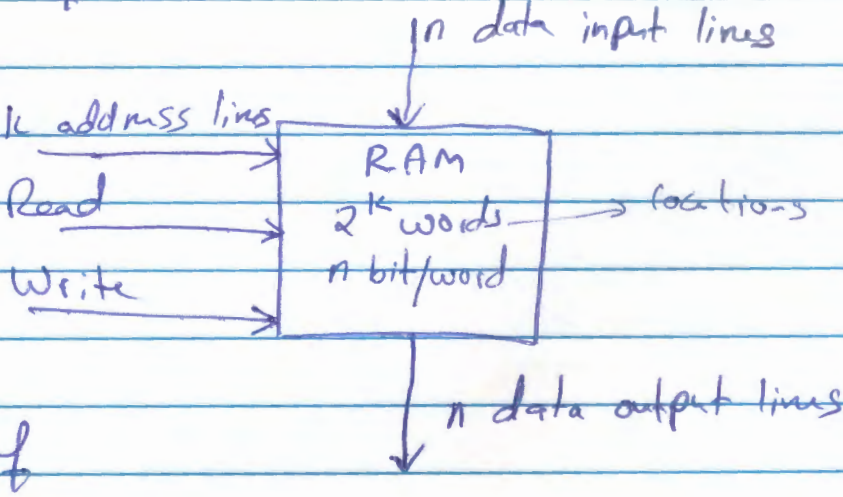
- memory unit stores binary information in groups of bits called words (word: is a group of bits. (~~15 and 0~~)
  e.g 8-bit byte
  16-bit 2 byte (word) — integer)
  32-bit 4byte (double word)

- capacity of a memory unit: either in bit or byte

- block diagram of RAM

- data input and output lines

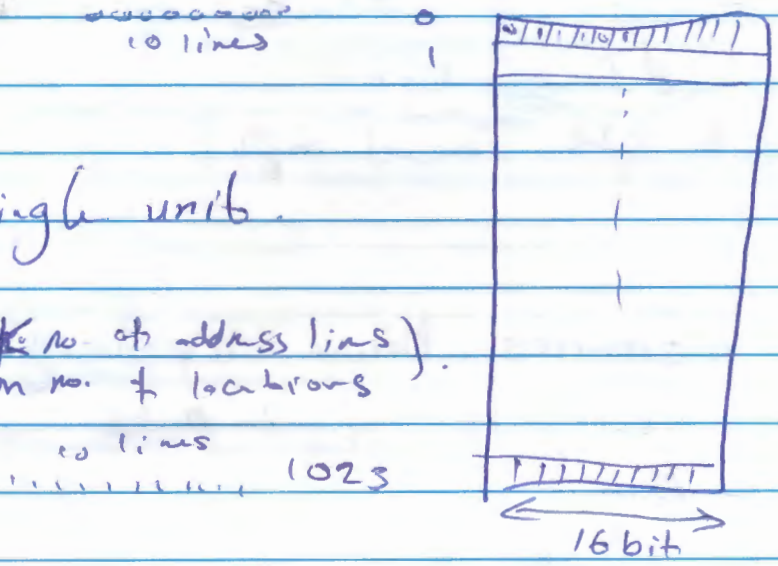
- Control lins:
  read and write.
  (the direction of data flow)



- address lins (k ~~lins~~ → 2^k locations)

- memory unit is specified by the number of words it contains and the number of bits in each word (eg. $2k \times 8$) → 2k byte or 16k bit
- each word has an address range of addresses $(0 - (2^k - 1))$.
- $1k = 2^{10}$ $1M = 2^{20}$ $1G = 2^{30}$
  $4k = 2^{12}$ $2G = 2^{31}$ ... etc.
- Example 1k words x 16 bit each
  ⟹ no. of address lines = 10

Capacity = 2k byte or 16k bit



- 16 bit as single unit.
- $2^k \geq n$ (k: no of address lines, n no. of locations)

Write and Read Operations

- write : transfer data to memory location
- read : " " out from " "

Write operation :

1) Apply the binary address of the desired word to address lines.
2) apply the data bits to the data input lines
3) activate the memory chip and write input

Read operation :

1. Apply the binary address of the desired word to the address lines
2. activate the read input.

Some memories have memory enable (chip select) input pin, and Read/write inputs are in the same pin

memory enable (CS)	Read/write	Memory operation
0	X	None
1	0	write operation
1	1	Read operation.

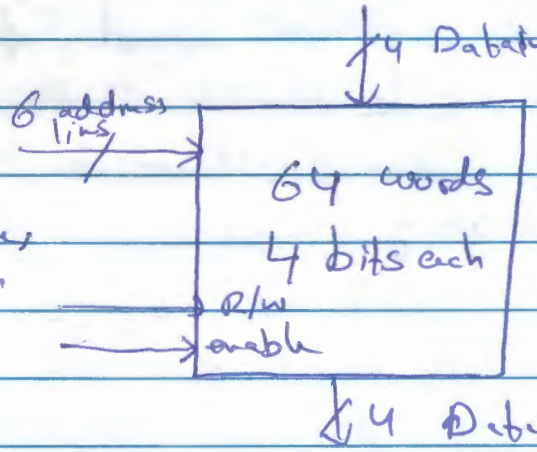
Memory Description in HDL

- Memory is modeled in Verilog HDL by an array of registers

```
reg [15:0] memword [0:1023]; // 2 dimension array
```

- this is 1k x 16 memory.
  (1024 registers, each containing 16 bits).
- now memword[7] refers to the 16-bit memory word at address 7.

Example




```
module memory(Enbly, ReadWrite,
  Address, DataIn, DataOut);
  input Enbly, ReadWrite;
  input [3:0] DataIn;
  input [5:0] Address;
  output [3:0] DataOut;
  reg [3:0] DataOut;
  reg [3:0] mem[0:63];
always @( Enbly or ReadWrite)
  if (Enbly == 0)
    DataOut = 4'bZ;
  else
    if (ReadWrite)
      DataOut = mem[Address];
    else
      mem[Address] = Data_In;
endmodule.
```

(X) Error Detection and Correction

(A) Some errors may occur in storing and retrieving the binary information.
(B) employ error detecting and correcting codes. (e.g parity bit is the most common).

(B) Hamming Code:

- In Hamming Code, K parity bits are added to an n-bit data word, forming a new word of n+K bits.
- The bit positions are numbered in sequence from 1 to n+k.
- Those positions numbered a a power of 2 are reserved for the parity bits. The remaining bits are the data bits.

~~- for n bits of data, we need~~

- The following relationship between n and k

$$2^k - 1 - k \geq n$$

e.g for $k = 3$ (check bits (parity bits)

$$n \leq 2^3 - 1 - 3 = 4$$

for $k = 4$

$$n \leq 2^4 - 1 - 4 = 11$$

$$\Rightarrow 5 \leq n \leq 11$$

if $n < 5 \rightarrow k = 3$ bits is enough.

Example: Consider the 8-bit data word 11000100

⇒ we need 4 parity bits

⇒ new word size = n + k = 8 + 4 = 12

Bit positions:

Bit positions:	1	2	3	4	5	6	7	8	9	10	11	12
	P_1	P_2	1	P_4	1	0	0	P_8	0	1	0	0

P_1 = XOR of bits (3, 5, 7, 9, 11)

= 1 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 0 = 0

P_2 = XOR of bits (3, 6, 7, 10, 11)

= 1 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 0 = 0

P_4 = XOR of bits (5, 6, 7, 12)

= 1 ⊕ 0 ⊕ 0 ⊕ 0 = 1

P_8 = XOR of bits (9, 10, 11, 12)

= 0 ⊕ 1 ⊕ 0 ⊕ 0 = 1

(EVEN PARITY).

3	0 0 1 1
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0

* Now the 8-bit data word is stored in memory together with the 4 parity bits as a 12-bits composite word.

Bit position:	1	2	3	4	5	6	7	8	9	10	11	12
	0	0	1	1	1	0	0	1	0	1	0	0

* when the 12 bits are read from memory, they are checked again for possible errors.

⟶

C_1 = XOR of bits (1, 3, 5, 7, 9, 11)
C_2 = XOR of bits (2, 3, 6, 7, 10, 11)
C_4 = XOR of bits (4, 5, 6, 7, 12)
C_8 = XOR of bits (8, 9, 10, 11, 12)

if $C = 0$ ⇒ Even parity ⇒ no errors
⇒ $C = C_8 C_4 C_2 C_1 = 0000$ indicates that no error in the stored word.
if $C \neq 0$ ⇒ an error has occurred

eg: 0 0 1 1 1 0 0 1 0 1 0 0 no error
⇒ $C_1 = 0$ $C_2 = 0$ $C_4 = 0$ $C_8 = 0$
⇒ No error.

1 0 1 1 1 0 0 1 0 1 0 0 error in bit 1
⇒ $C_1 = 1$ $C_2 = 0$ $C_4 = 0$ $C_8 = 0$
⇒ error in bit 1 (the parity bit).

0 0 1 1 0 0 0 1 0 1 0 0 error in bit 5
⇒ $C_1 = 1$ $C_2 = 0$ $C_4 = 1$ $C_8 = 0$
⇒ error in bit (4+1) = 5

- C is called the syndrome.
- The syndrome value C consists of k bits and has a range of 2^k values between 0 and $2^k - 1$.
- The value $C = 0$ indicates no error ⇒ $2^k - 1$ are left to indicate which

of the $n+k$ bits was in error.

$\Rightarrow 2^k - 1 \geq n+k$

$\Rightarrow \underline{2^k - 1 - k \geq n}$

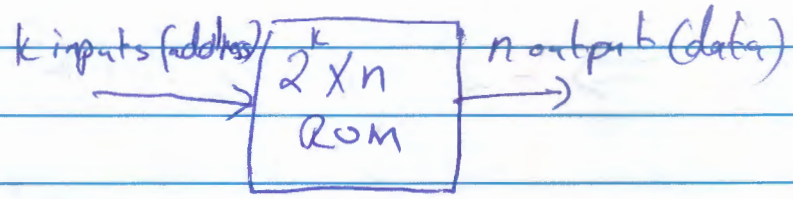
- Up to now we can detect one error and correct one error

- Singh error Correction, Double error Detection
  - By adding another parity bit to the coded word, the hamming code can be used to correct a single error and detect double errors

- in the previous example
  $0\,0\,1\,1\,1\,0\,0\,1\,0\,1\,0\,0\,P_{13}$
  P_3 = XOR of all pourious bits
  $\Rightarrow P_3 = 1$ (even parity).
- thus 13 bits are stored in memory
- when reading data P= XOR of the 13-bits
  - if $c = 0$ and $P = 0 \Rightarrow$ no error occured
  - if $c \neq 0$, $P = 1$, a singh error has occured that can be corrected
  - if $c \neq 0$, $P = 0$, a double error has occured that can not be corrected.
  - if $c = 0$, $P = 1 \Rightarrow$ An error occured in the P13 bit.

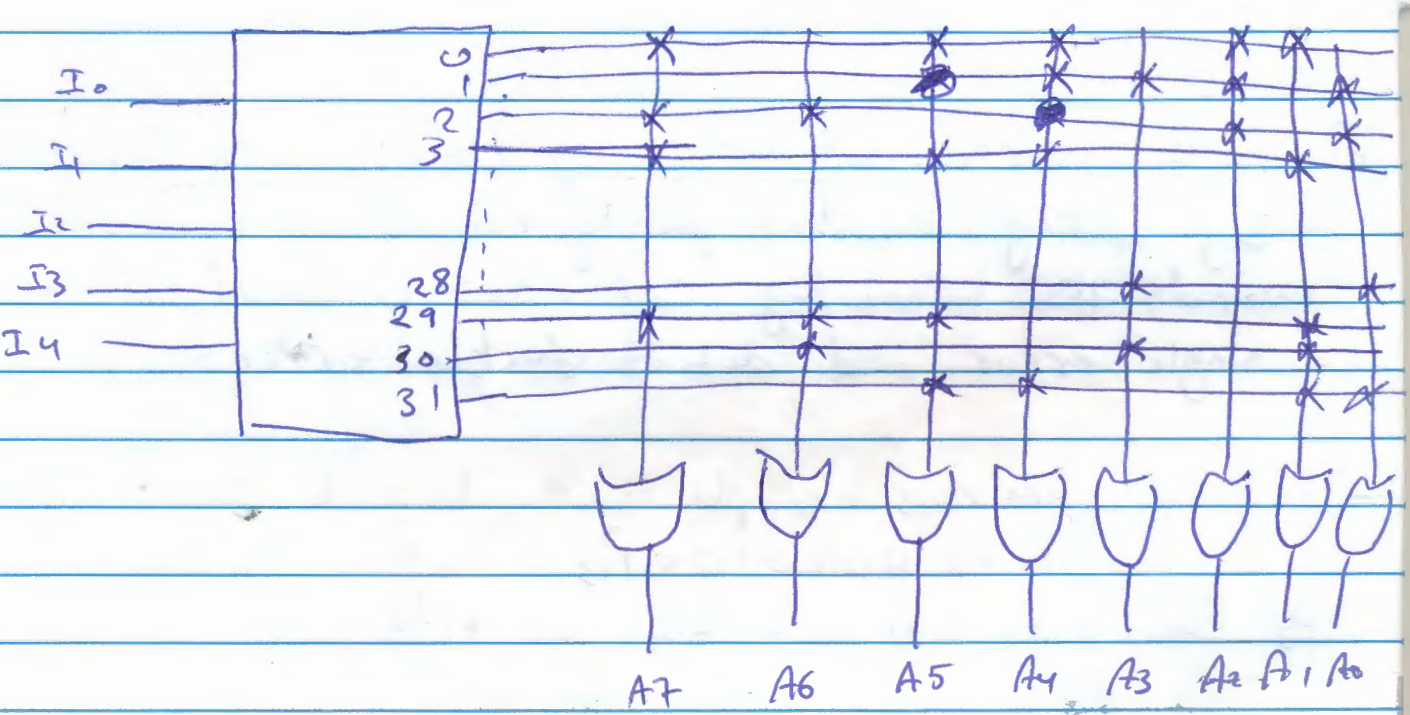
ROM:

- Read only
- nonvolatile memory



- internal block diagram

Eg. 32×8 ROM (5 address lines, 8 output lines)



- 8 OR gates with 32 inputs for each.

$\Rightarrow$ this ROM has $8 \times 32 = 256$ internal connections.

- In general, $2^k \times n$ ROM will have an internal $k \times 2^k$ decoder and n OR gates. Each OR gate has 2^k inputs.

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	0	0	0	1	0	0	1
0	0	0	1	1	0	1	1	0	0	1	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	0	1	0	0
1	1	1	1	1	0	0	1	1	0	0	1	1

* Combinational Circuit Implementation
* ROM contains Decoder & OR gates.
  ⇒ ROM outputs can be programmed to represent the boolean functions.

* ROM can be interpreted in two ways
  1. memory unit that contains a fixed pattern of stored word
  2. unit that implements a combinational circuit.

  e.g. $A_7(I_4, I_3, I_2, I_1, I_0) = \sum(0, 2, 3, \ldots, 29)$

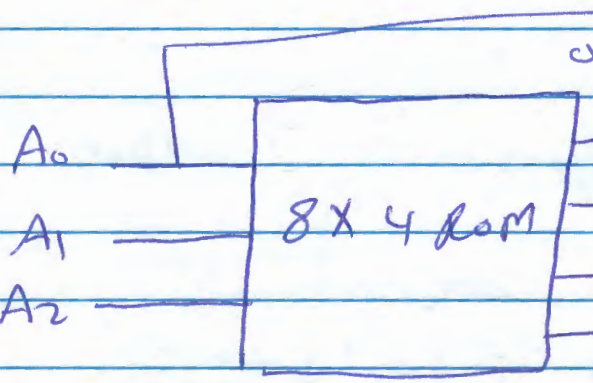
Example:-

3-bit number, output equal the square of the input number. Implement this function using ROM

⇒ 6 outputs

Inputs			Outputs					
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	0	0
0	1	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0	1
1	1	0	1	0	0	1	0	0
1	1	1	1	1	0	0	0	1

$B_0 = A_0$ $B_1 = 0$



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

Types of ROM

1) mask programming : programming done during the fabrication process according to a truth table by the user (customer)
2) PROM : programmed by the user for one time.
3) EPROM : Erasable PROM through a special device using ultraviolet radiation for a period of time
4) EEPROM : Electrially Erasable PROM

Combinational PLD

- 3 main types : differ in the placement of the programmable connections in the AND-OR array

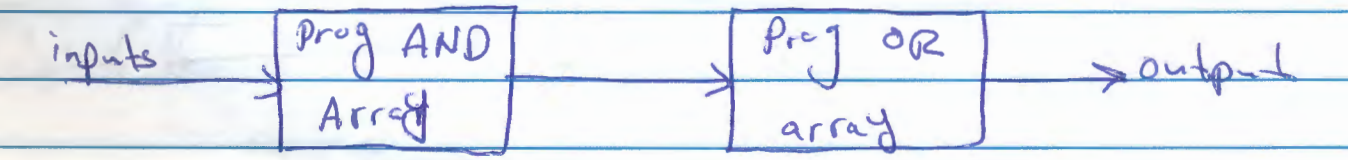
(1)

(PROM - Prog. read only memory)

(2)

Programmable Array Logic (PAL)

(3)



Programmable Logic Array (PLA)

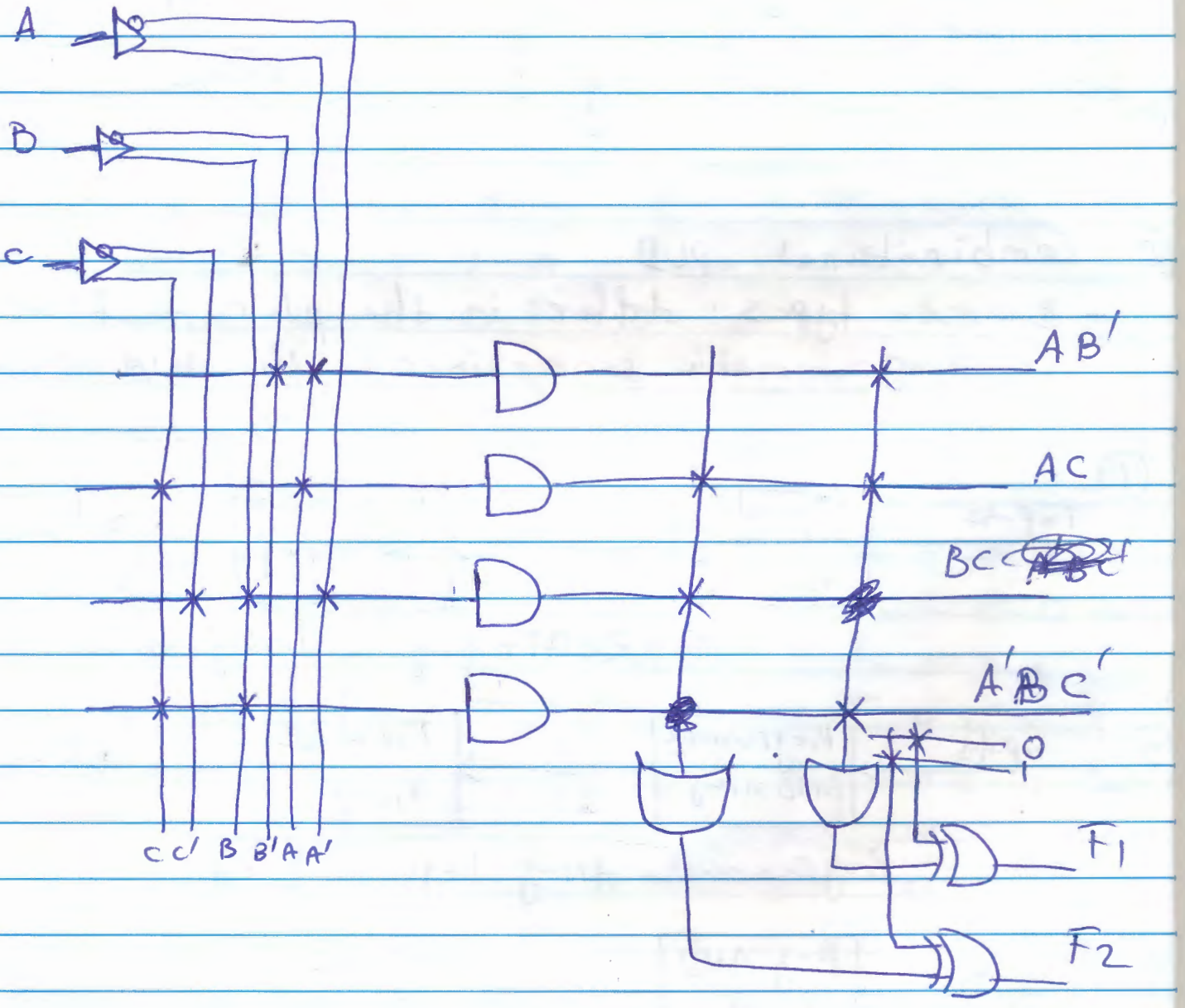
⊛ Programmable Logic Array (PLA)

- No decoder → doesn't generate all minterms like PROM.

Example: PLA with 3 inputs and 2 outputs

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

$F_2 = (AC + BC)'$



⊛ synaptyCad

⊛ Programming Table

Product Term		Inputs A	B	C	outputs (T) F1	(C) F2
AB'	1	1	0	–	1	–
AC	2	1	–	1	1	1
~~ABC~~ BC	3	–	1	1	–	1
~~BC~~ $A'BC'$	4	0	1	0	1	–

↓ no need for this

Example: Implement the following two Boolean functions with a PAL

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

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

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

$F_1 = A'B' + A'C' + B'C'$

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

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

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

$F_2 = (A'C + A'B + AB'C')'$

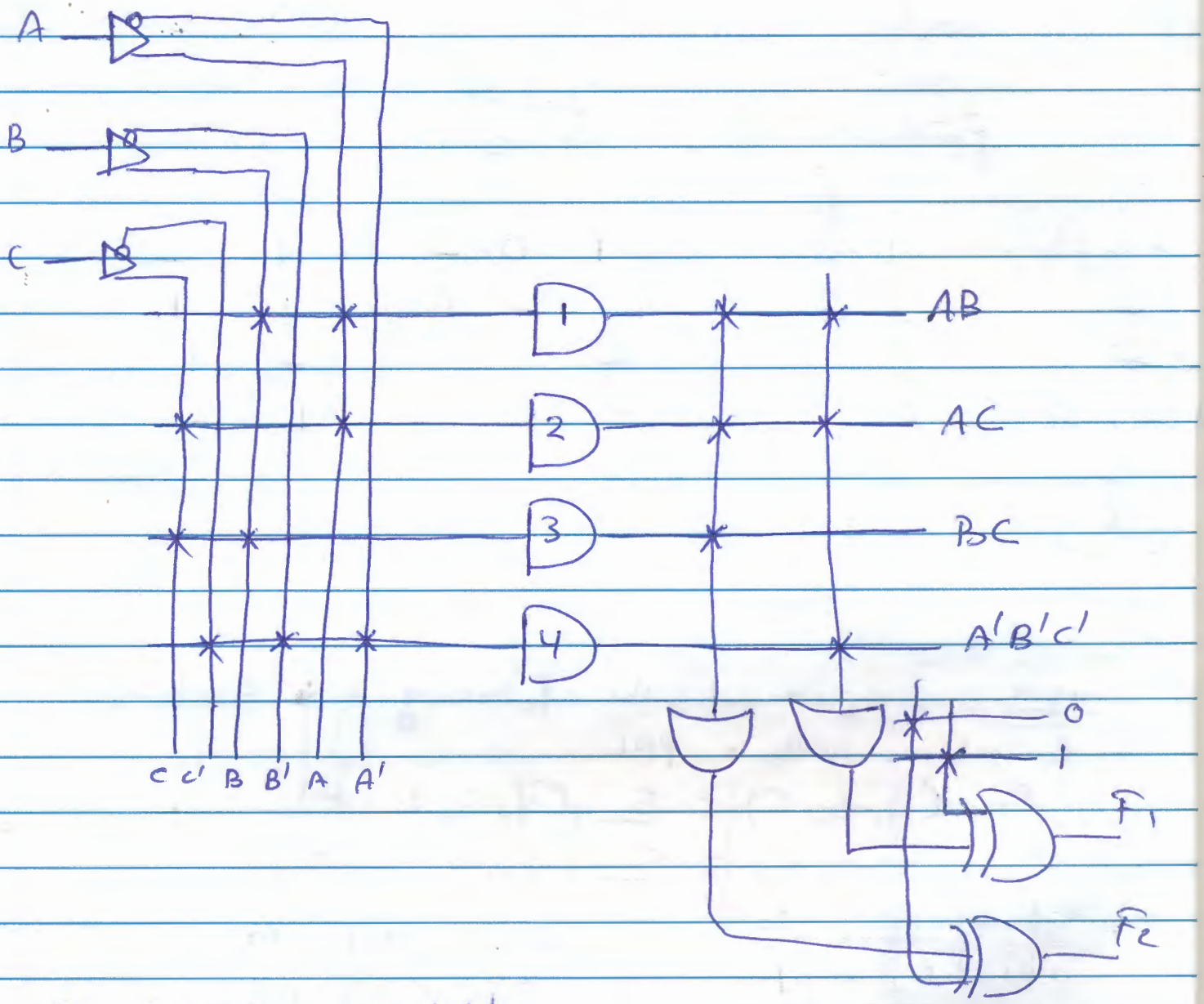
$F_1(T)$ & $F_2(T) \Rightarrow$ 6 terms

$F_1(T)$ & $F_2(C) \Rightarrow$ 6 terms

$F_1(C)$ & $F_2(T) \Rightarrow$ 4 terms

$F_1(C)$ & $F_2(C) \Rightarrow$ 6 terms

⟹



* Programming table

	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

- In general, For n inputs, k product terms and m outputs the internal logic of the PLA

consists of n-buffer-inverter gates, K AND gates, m OR gates, and m XOR gates. There are $2n \times k$ connections between the inputs and the AND array, $k \times m$ connections between the AND and OR arrays and m connections with the XOR gates.

Symptycal