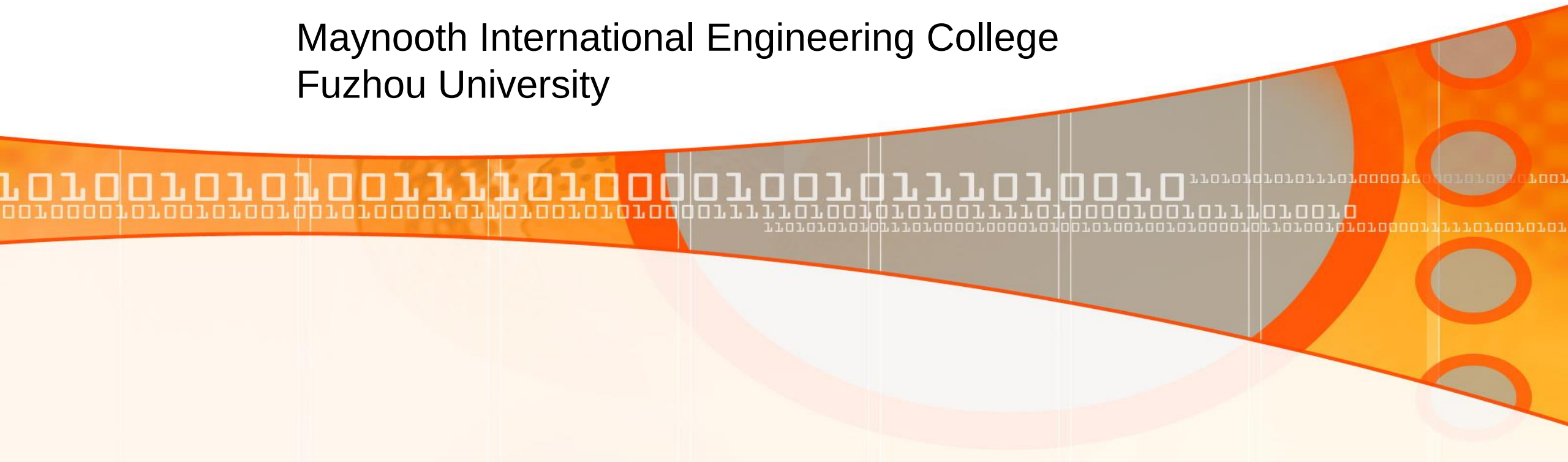
EE103 Digital Systems 1

Wong Chin Hong

106

Maynooth International Engineering College
Fuzhou University



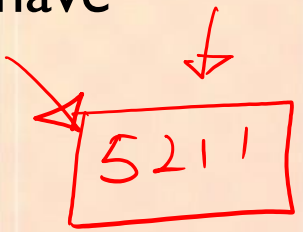
So far ... !

- We've used Karnaugh Maps to minimise logic ...
- We've implemented circuits using NAND only or NOR only gates ...
- We've analysed and designed a counter ...
- We've carried out multi-output minimisation and looked at the seven segment display …

TODAY, we are going to look at the FINAL section of the notes – namely Programmable Logic Devices

Programmable Logic Devices

- So far in this module (and particularly in our laboratories), we have implemented digital circuits using **integrated circuits** (ICs).



- As we know, the IC (often referred to as a chip) contains the electronic components for the digital gates and storage elements. The various components are interconnected in the chip.

- The number of pins on a chip can vary from 14 on a small IC package (as we have used in our labs) to several hundred on a large package.

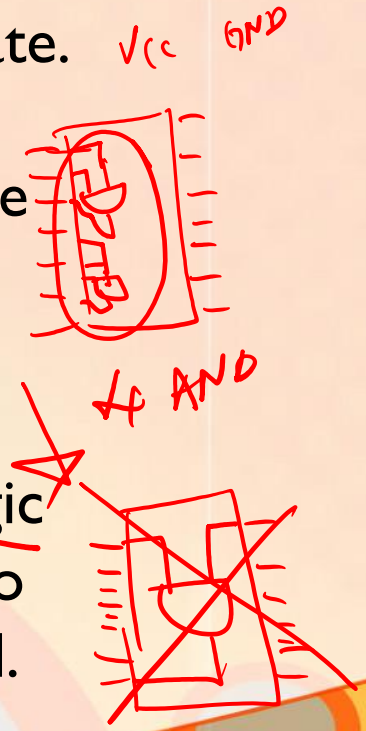
- Each IC has an alphanumeric identifier printed on its surface and typically has an associated datasheet containing all its relevant information (such as pin layout etc.). These datasheets are typically found on the manufacturer's website.

Programmable Logic Devices

- Implementation technology using and connecting together ICs is regarded as **fixed** in the sense that the hardware functionality has been specified at the manufacturing stage. A CMOS 4081 (AND) IC, for example, will only implement the logic operation of an AND gate.

- An alternative implementation technology, known as programmable logic devices (PLDs) have undefined functions at the time of manufacture.

- They are fabricated with structures that can implement various logic functions and structures that are used to control connections or to store information specifying the actual logic functions implemented.



Programmable Logic Devices

- Such devices require **programming (or reconfiguring)** post manufacturing. Programming relates to making the necessary hardware connections to implement relevant functions.

- These connections can consist of **fuses**, which offer a once-off solution, or, more commonly, they can consist of metal oxide semiconductor field effect transistors (MOSFETs).

- **MOSFETs offer an erasable solution**, i.e. can be programmed, erased and reprogrammed as needed. These will be studied in detail in later relevant modules.

Programmable Logic Devices

- In this section of the notes, we are going to examine three types of basic programmable logic devices, namely the **Read Only Memory (ROM)**, the **Programmable Array Logic (PAL)** and the **Programmable Logic Array (PLA)**.

ROM
PAL
PLA

- More advanced programmable devices, such as field-programmable gate arrays (FPGAs), will be studied in later modules.

- ROM, PAL and PLA all have similar structures but differ in their programmability.

- As a result, they have slightly different requirements on the minimisation process needed to implement a set of logic functions. These will be outlined in the next sections.

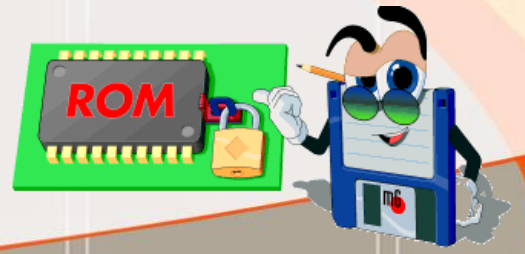
Read Only Memory (ROM)

- Consider the implementation of the following multi-output functions:

$$f_{1(A1, A0)} = \sum(1, 2, 3)$$
$$f_{2(A1, A0)} = \sum(0, 2)$$

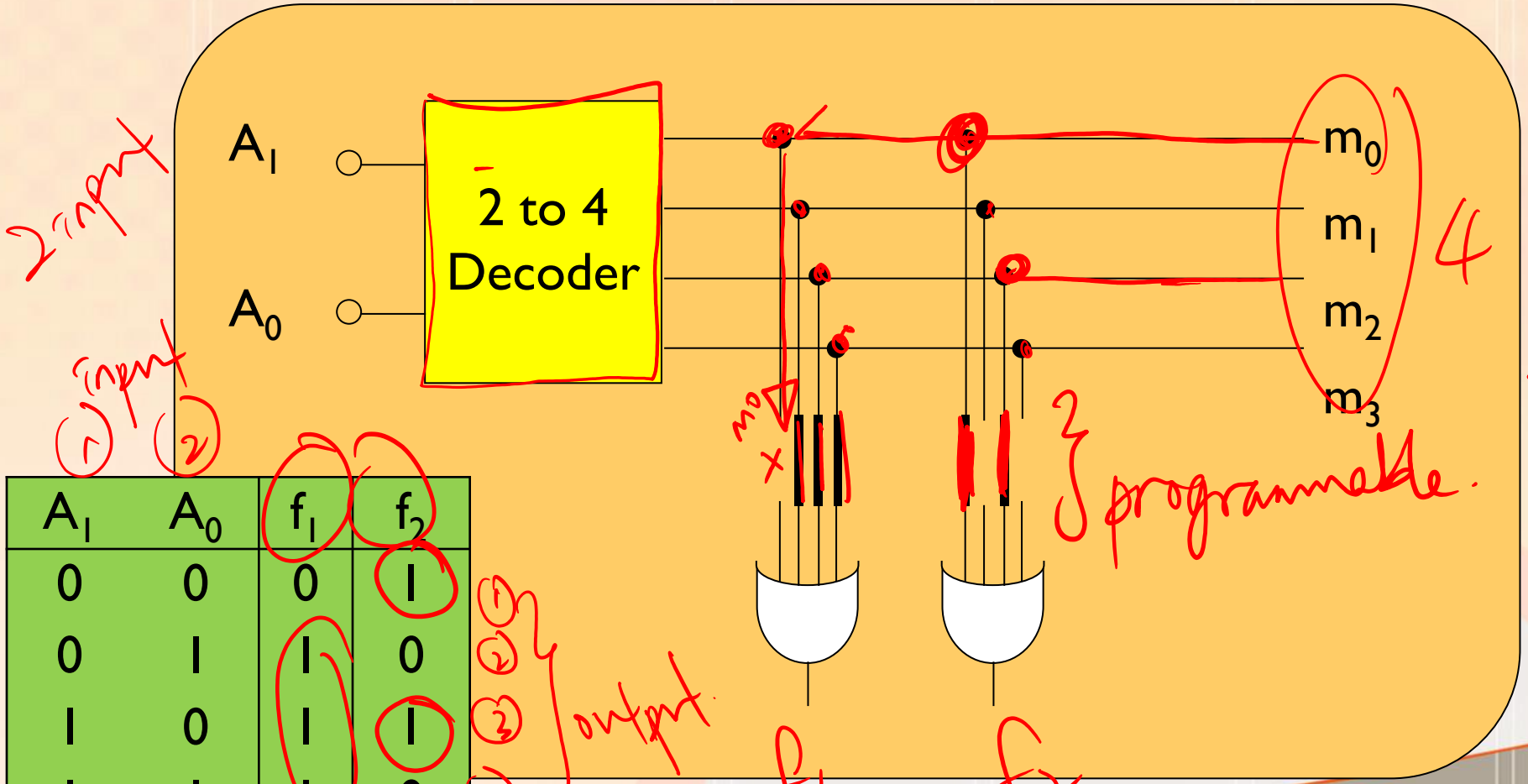
- In tabular form we can express the functions as follows:

	A_1	A_0	f_1	f_2
m_0	0	0	0	1
m_1	0	1	1	0
m_2	1	0	1	1
m_3	1	1	1	0



Read Only Memory (ROM)

- A simple conceptual circuit to implement these functions would be:

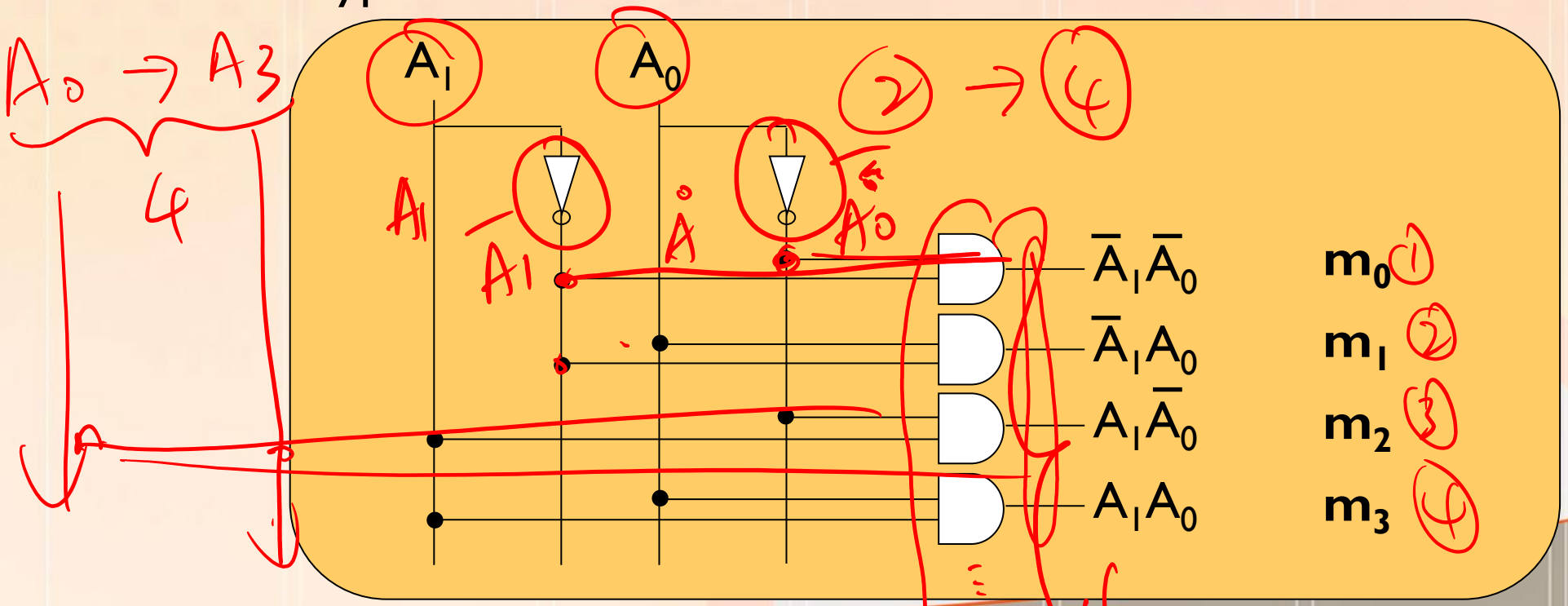


	A_1	A_0	f_1	f_2
m_0	0	0	0	1
m_1	0	1	1	0
m_2	1	0	1	1
m_3	1	1	1	0

Read Only Memory (ROM)

- The decoder is a standard circuit that produces all combinations of a given set of inputs. Here, we have 2 inputs and hence $2^2 = 4$ possible outputs, i.e. the minterms.

- The typical 2 to 4 decoder circuit is:

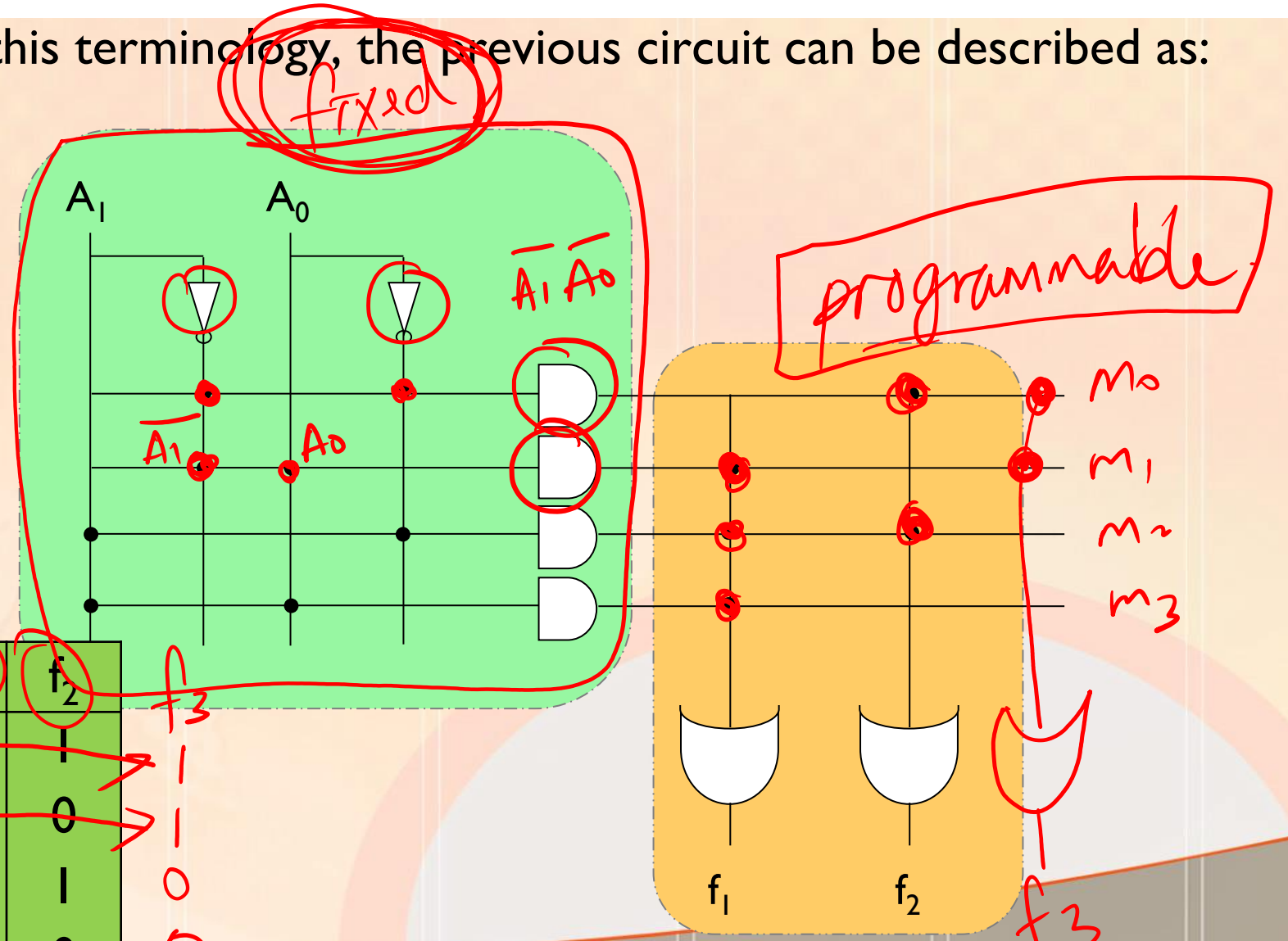


Read Only Memory (ROM)

- For convenience, we use the following alternative representation:

Read Only Memory (ROM)

- Using this terminology, the previous circuit can be described as:



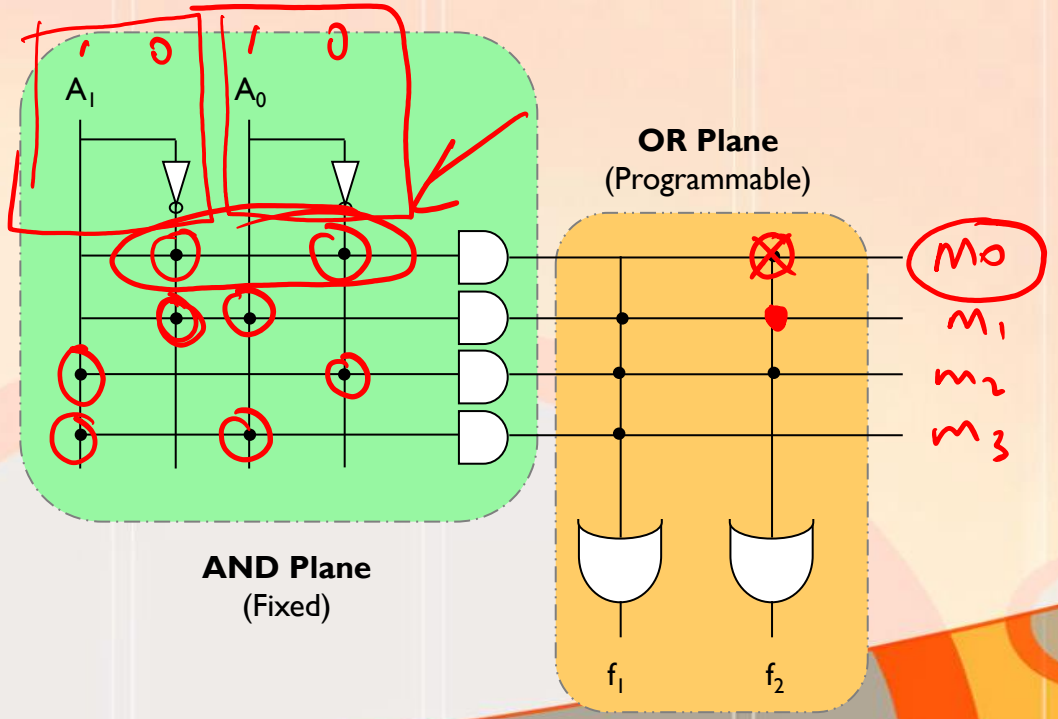
A_1	A_0	f_1	f_2
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

m_0
m_1
m_2
m_3

f_3
1
1
0
0

Read Only Memory (ROM)

- This circuit is, in effect, a programmable **ROM** (or PROM).
- This structure has two distinct sections – the AND plane and the OR plane. This is the same basic structure for the PAL and PLA devices also.
- Here, the AND plane is fixed, i.e. it is not programmable. All combinations of the inputs are produced as outputs of the AND plane, irrespective of whether or not they are needed.

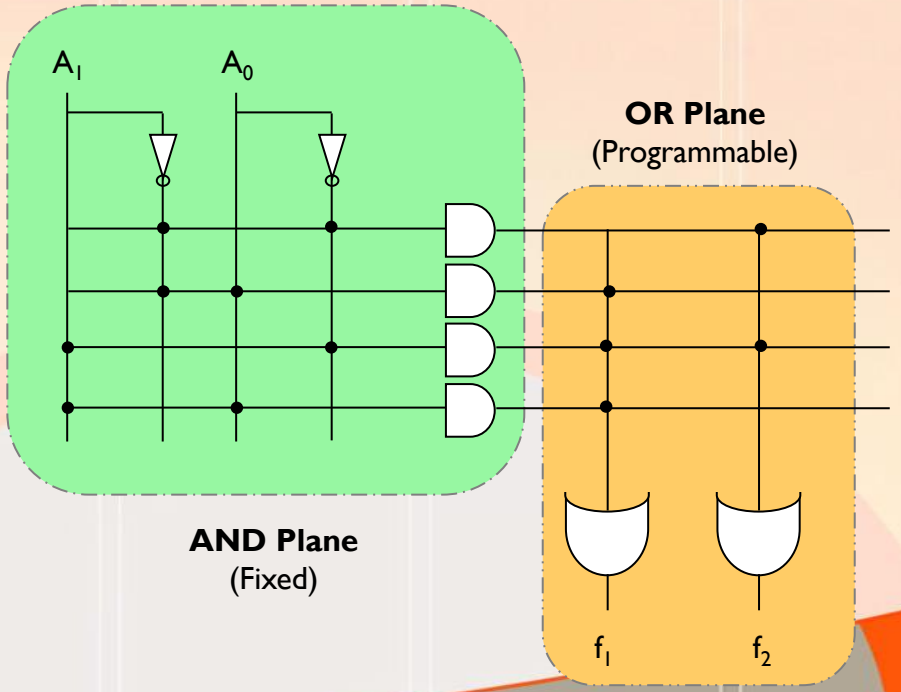


Read Only Memory (ROM)

- The OR plane, on the other hand, is programmable and we can program a particular function based on the appropriate selection of minterms.

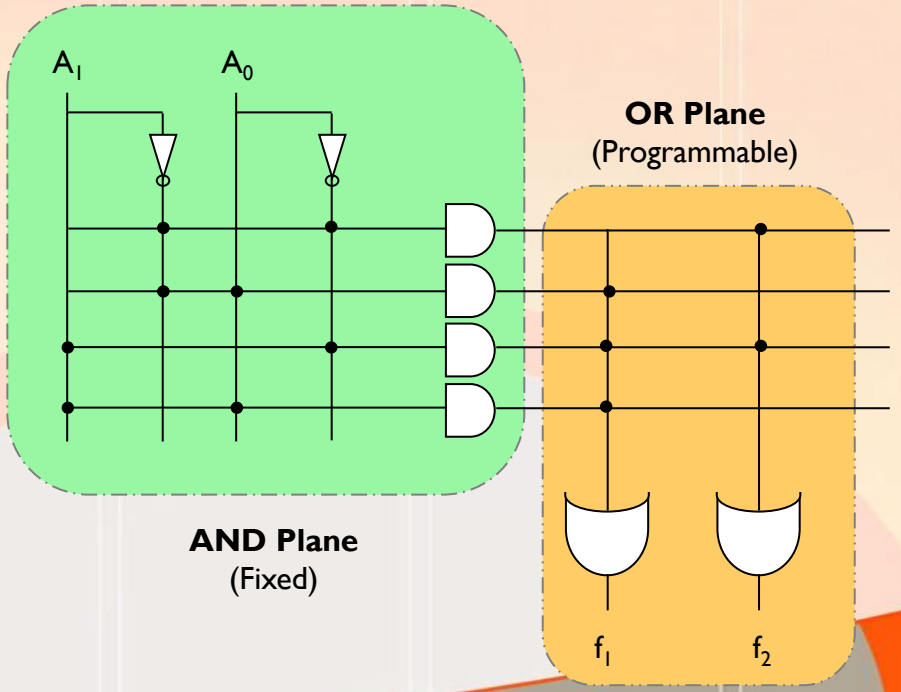
- The functionality of the ROM can be viewed in two different ways.

- Firstly, **it can be viewed as implementing multiple functions**, as we have just seen. The inputs are A_1 and A_0 and f_1 is the first ROM output and f_2 is the second ROM output.



Read Only Memory (ROM)

- The OR plane, on the other hand, is programmable and we can program a particular function based on the appropriate selection of minterms.
- The functionality of the ROM can be viewed in two different ways.
- It **can also be viewed as a means of storing code**.
- In this scenario, the inputs A_0 and A_1 are regarded as **address lines** and the **outputs are words** consisting of bits from f_1 and f_2.



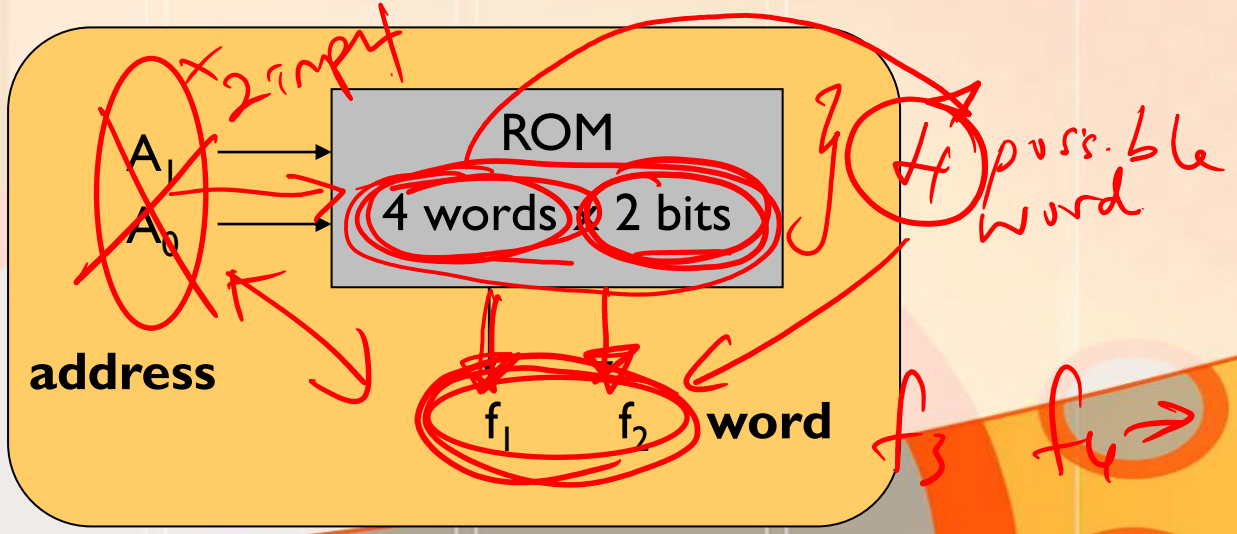
Read Only Memory (ROM)

- For example, when **address** $A_1A_0 = 00$, the output word is $f_1f_2 = 01$.

	A_1	A_0	f_1	f_2
m_0	0	0	0	1
m_1	0	1	1	0
m_2	1	0	1	1
m_3	1	1	1	0

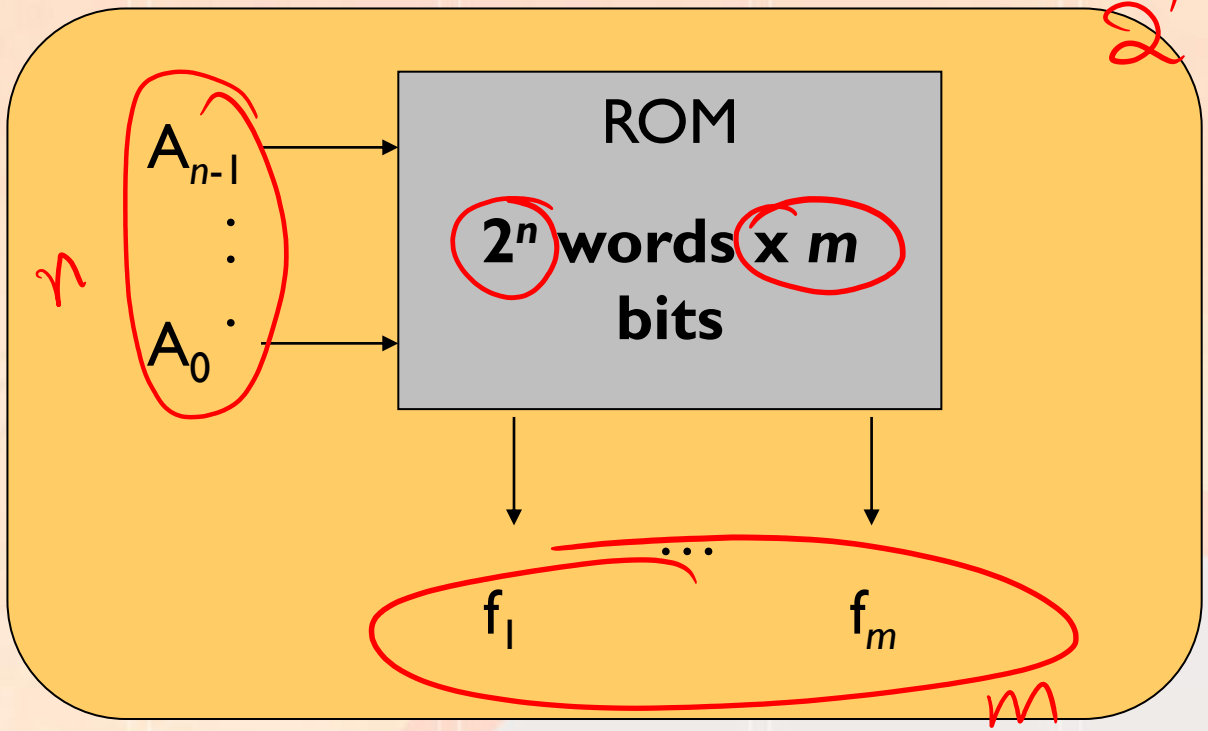
- Here, as we have 2 inputs, there are 4 possible **words** produced.

- The size of each word depends on the number of ROM outputs.



Read Only Memory (ROM)

- In general, a ROM with *n* input lines and *m* output lines contains an array of **2^n words each of *m* bits**:



Read Only Memory (ROM)

- ***Ex. 9.1 Implement the multi-output circuit represented by the following truth table on a ROM structure:***

3 input 4 output

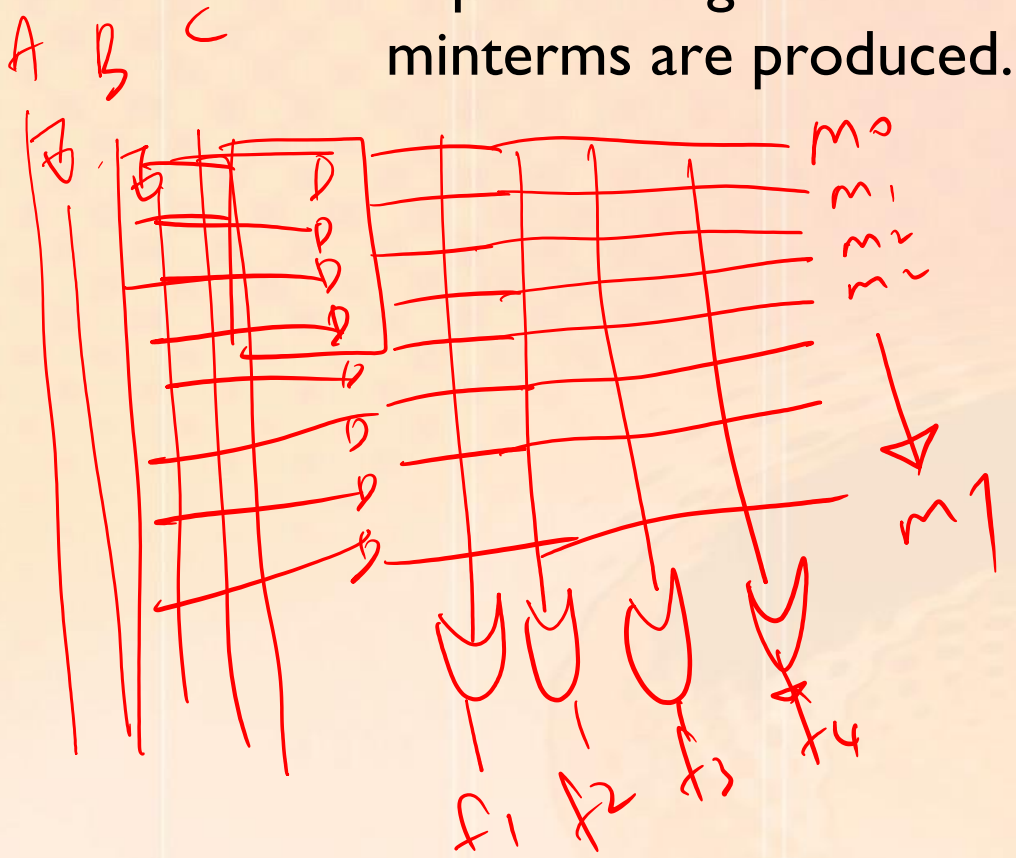
$2^3 \times 4$

$= 32$

A	B	C	f_1	f_2	f_3	f_4
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	1	0	0	0	1	0
1	1	1	0	0	0	0

Read Only Memory (ROM)

- Implementing this on a **ROM** requires a **3 to 8 decoder**, i.e. 8 minterms are produced.



8 minterms

A	B	C	f_1	f_2	f_3	f_4
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	1	0	0	0	1	0
1	1	1	0	0	0	0

Read Only Memory (ROM)

- Implementing this on a **ROM** requires a **3 to 8 decoder**, i.e. 8 minterms are produced.

- The OR plane is then programmed by choosing the appropriate minterms for each of the four functions as per the table. Note, **there is no minimisation necessary**.

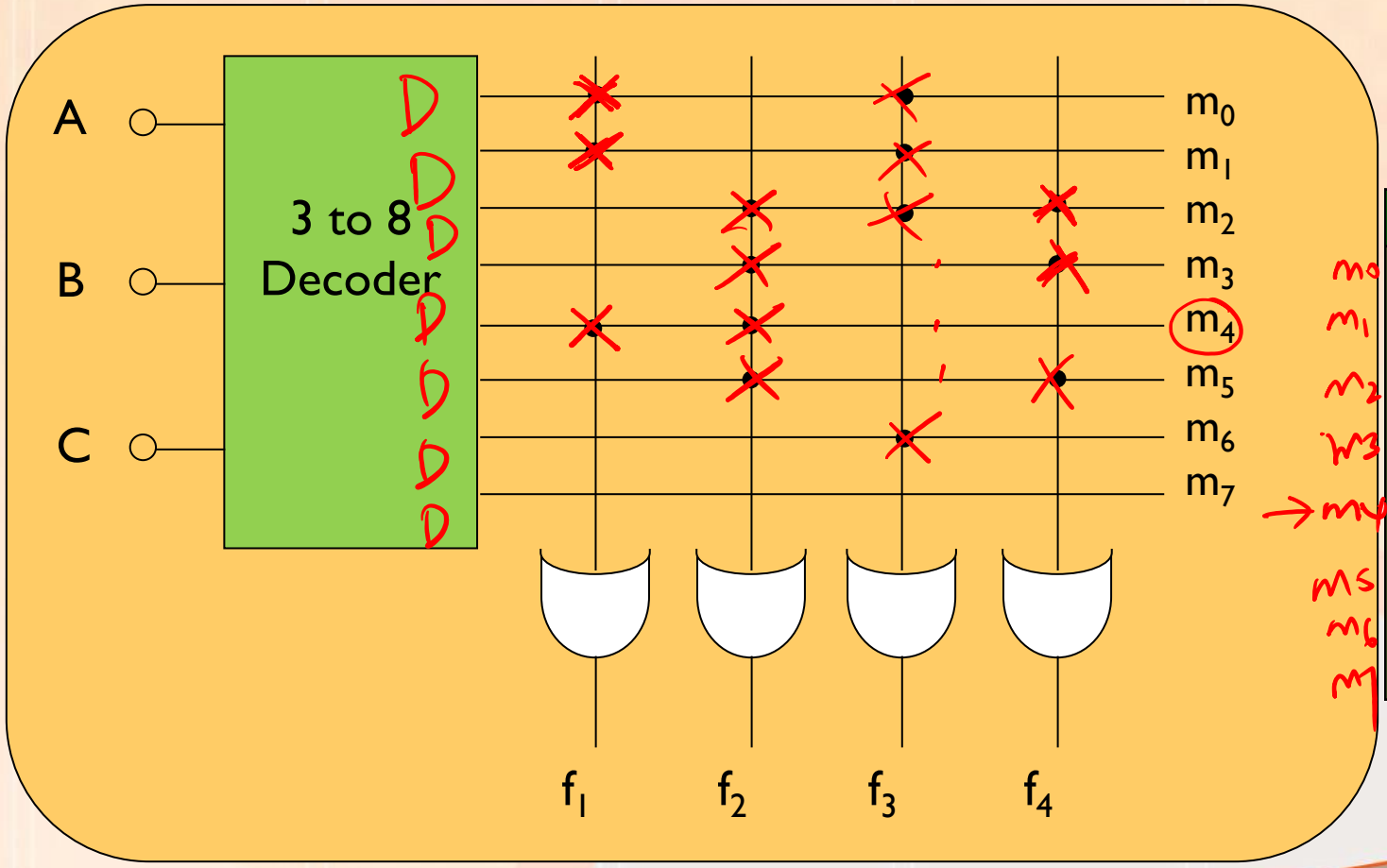
- The size of the ROM is 8 words x 4 bits.

8 minterms

A	B	C	f_1	f_2	f_3	f_4
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	1	0	0	0	1	0
1	1	1	0	0	0	0

Read Only Memory (ROM)

- The ROM implementation is:



A	B	C	f_1	f_2	f_3	f_4
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	1	0	0	0	1	0
1	1	1	0	0	0	0

Programmable Logic Array (PLA)

- A Programmable Logic Array (PLA) comprises a programmable **AND plane connected to a programmable OR plane**.
- It does not decode the AND plane and, therefore, does not provide all possible minterms. Instead, it allows the AND plane to be programmed so as to generate product terms of the inputs variables.
- In the case of the PLA structure, we need to first minimise each of the output functions in order to obtain a minimal set of produce terms and, hence, a reduced AND plane.
- Furthermore, since the OR plane is also programmable and the functions can share the product terms from the AND plane, we need to **carry out multi-output minimisation**.

Programmable Logic Array (PLA)

- ***Ex. 9.2 Implement the multi-output circuit given in Ex. 9.1 on a PLA structure.***

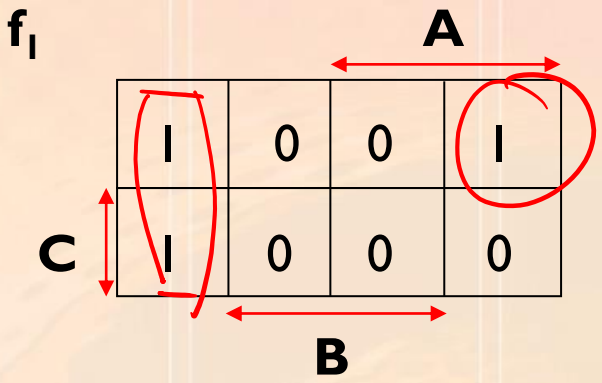
- ***Recall:***

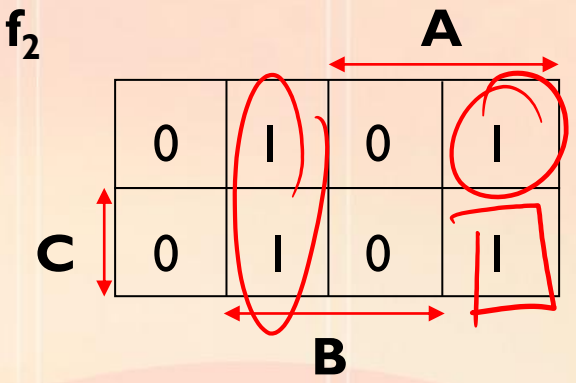
A	B	C	f_1	f_2	f_3	f_4
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	1	0	0	0	1	0
1	1	1	0	0	0	0

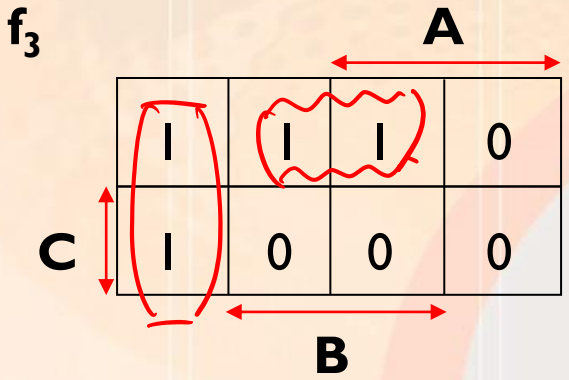
PLA → constraint
K-Map
↓
minimized
Multi-output

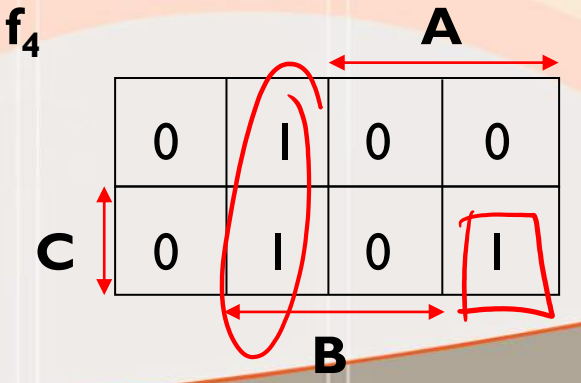
Programmable Logic Array (PLA)

- Before we can implement the circuit, we need to first **use Karnaugh Maps and multi-output minimisation** in order to obtain the least number of product terms in the AND plane.









Programmable Logic Array (PLA)

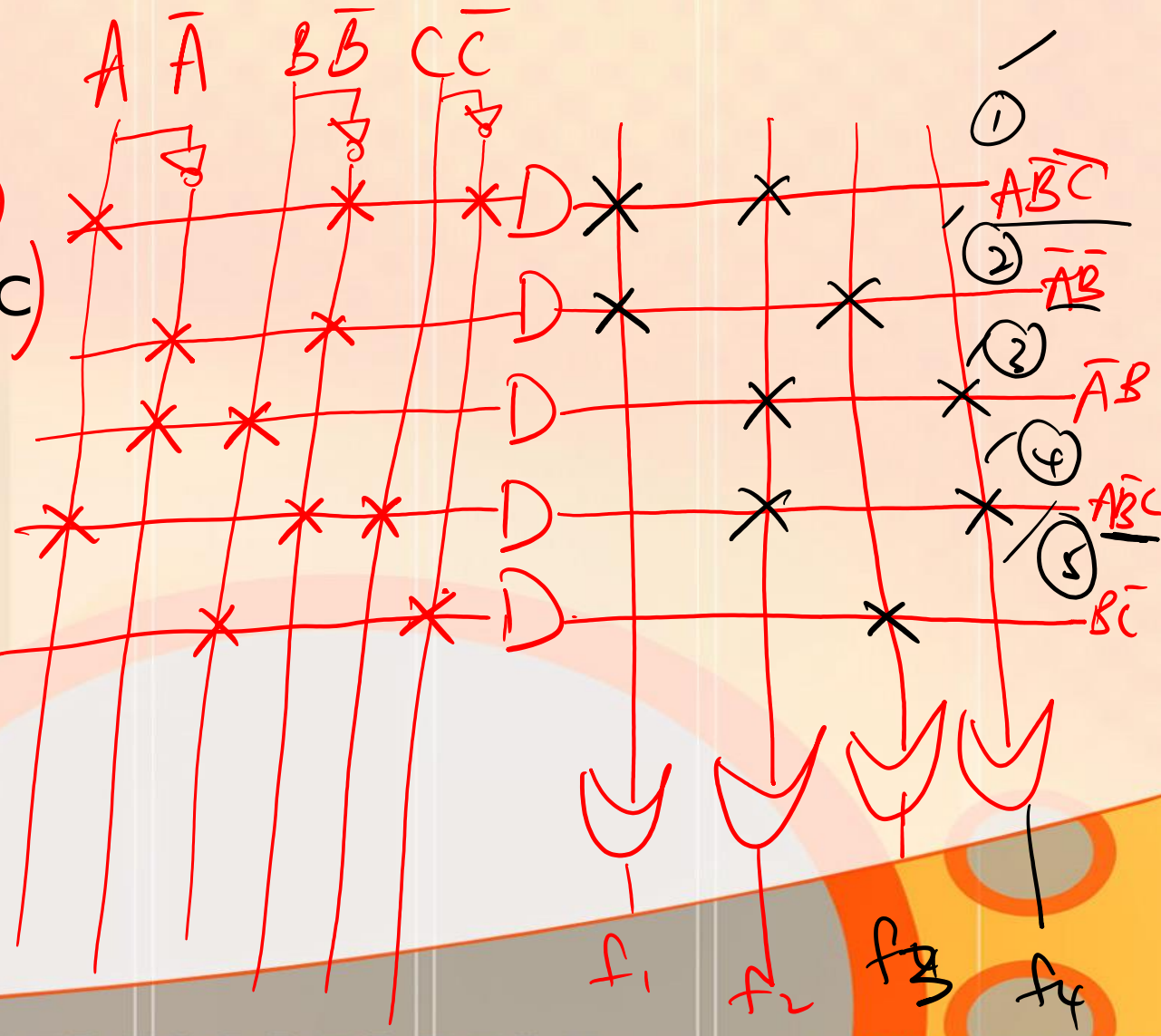
- This gives:

$f_1 = \bar{A}\bar{B}\bar{C} + A\bar{B}$

$f_2 = \bar{A}B + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C}$

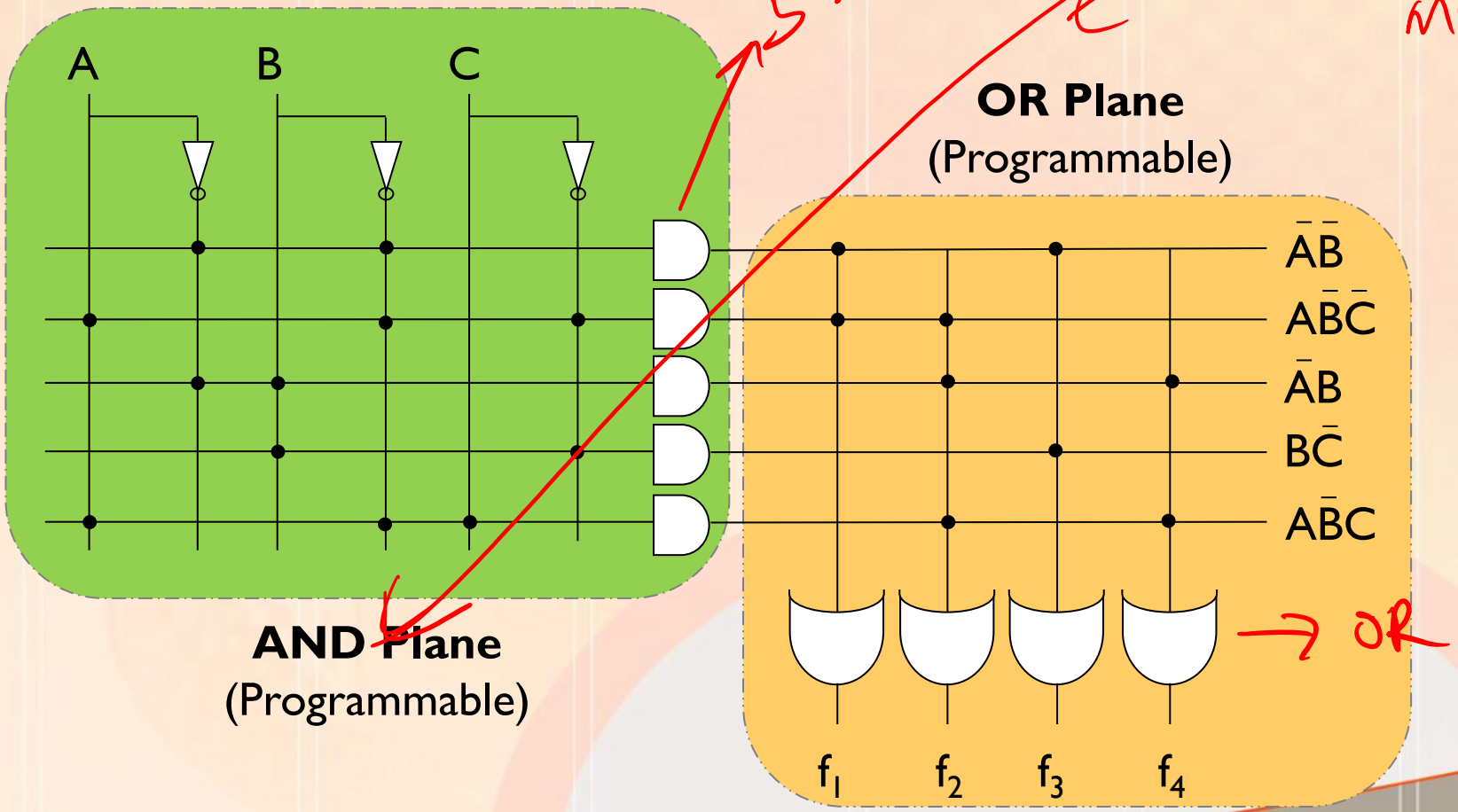
$f_3 = A\bar{B} + B\bar{C}$

$f_4 = \bar{A}B + \bar{A}\bar{B}\bar{C}$



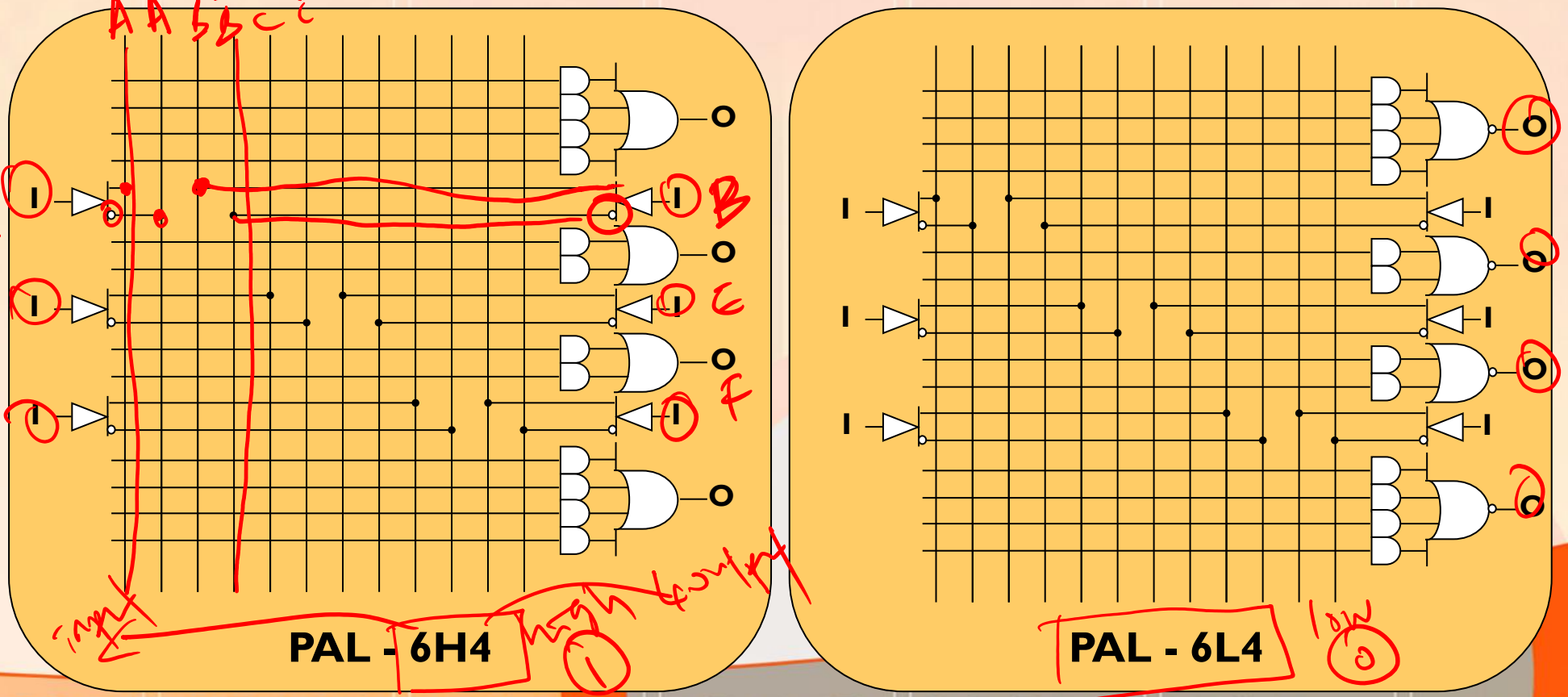
Programmable Logic Array (PLA)

- The PLA implementation is thus:



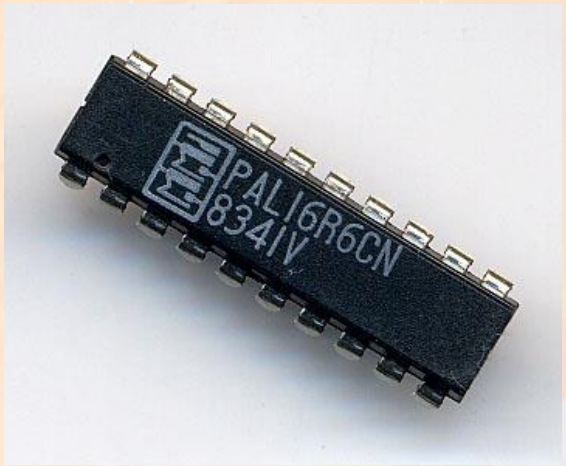
Programmable Array Logic (PAL)

- A Programmable Array Logic (PAL) represents the third variation of the programmable logic devices where the **OR matrix is fixed**.
- Two sample PAL structures are shown below:



Programmable Array Logic (PAL)

Hence, **for PAL**, we only need to **minimise each function individually.**



Programmable Array Logic (PAL)

- ***Ex. 9.3 Implement the multi-output circuit given in Ex. 9.1 on a PAL 6H4 structure.***

- ***Once again:***

A	B	C	f_1	f_2	f_3	f_4
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	1	0	0	0	1	0
1	1	1	0	0	0	0

Programmable Array Logic (PAL)

- Before we can implement the circuit, we need to first **use Karnaugh Maps and individually minimise** each of the four functions.

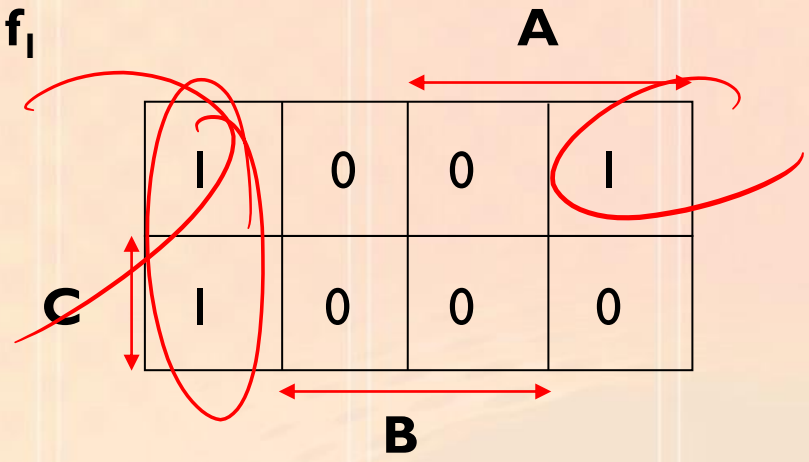
- Since we are using the **PAL 6H4** structure, we **need to group the 1's in the Karnaugh Maps**.

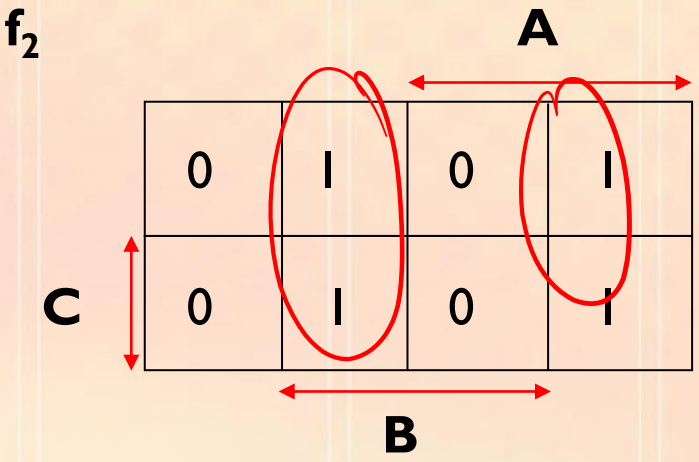
A	B	C	f_1	f_2	f_3	f_4
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	1	0	0	0	1	0
1	1	1	0	0	0	0

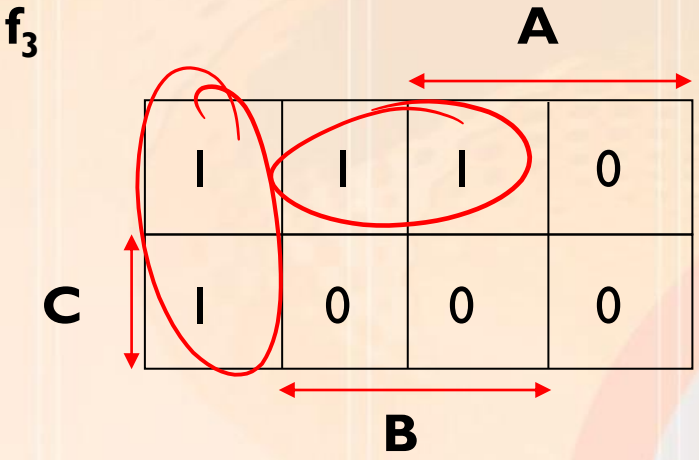
Group 1's

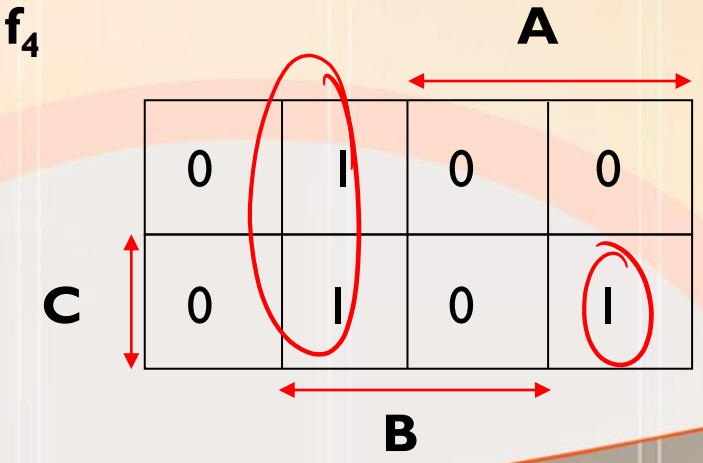
Karnaugh Maps - individual

Programmable Array Logic (PAL)









Programmable Array Logic (PAL)

- This gives:

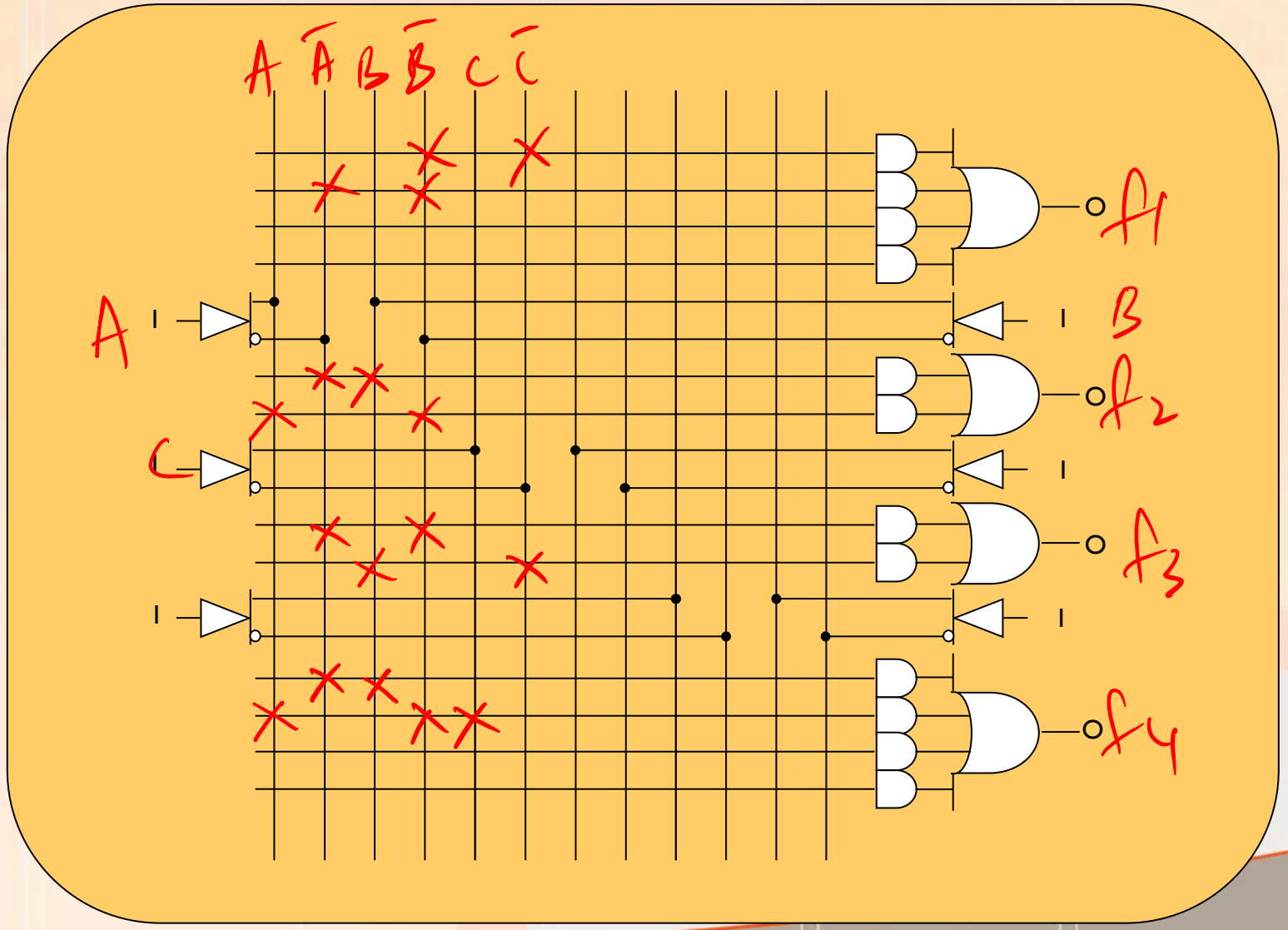
$$f_1 = \overline{B}\overline{C} + \overline{A}\overline{B}$$

$$f_2 = \overline{A}B + A\overline{B}$$

$$f_3 = \overline{A}\overline{B} + B\overline{C}$$

$$f_4 = \overline{A}B + AB\overline{C}$$

Programmable Array Logic (PAL)



$f_1 = \bar{B}\bar{C} + \bar{A}\bar{B}$

$f_2 = \bar{A}B + A\bar{B}$

$f_3 = \bar{A}\bar{B} + B\bar{C}$

$f_4 = \bar{A}B + A\bar{B}C$