Computer Logic Design Fundamentals

Chapter 5 – Digital Hardware Implementation

Part 2 – Programmable Implementation Technologies

Prof. Yueming Wang
ymingwang@zju.edu.cn
College of Computer Science and Technology,
Zhejiang University

Overview

- Part 1 – The Design Space
- Part 2 - Programmable Implementation Technologies
  - Why Programmable Logic?
  - Programming Technologies
  - Read-Only Memories (ROMs)
  - Programmable Logic Arrays (PLAs)
  - Programmable Array Logic (PALs)

Why Programmable Logic?

- **Facts:**
  - **It is most economical to produce an IC in large volumes**
  - **Many designs required only small volumes of ICs**
- **Need an IC that can be:**
  - **Produced in large volumes**
  - **Handle many designs required in small volumes**
- **A programmable logic part can be:**
  - **made in large volumes**
  - **programmed to implement large numbers of different low-volume designs**

Programmable Logic - More Advantages

- **Many programmable logic devices are *field-programmable*, i. e., can be programmed outside of the manufacturing environment**
- **Most programmable logic devices are *erasable* and *reprogrammable*.**
  - **Allows "updating" a device or correction of errors**
  - **Allows reuse the device for a different design - the ultimate in re-usability!**
  - **Ideal for course laboratories**
- **Programmable logic devices can be used to prototype design that will be implemented for sale in regular ICs.**
  - **Complete Intel Pentium designs were actually prototyped with specialized systems based on large numbers of VLSI programmable devices!**

Programming Technologies

- Programming technologies are used to:
  - Control connections
  - Build lookup tables
  - Control transistor switching
- The technologies
  - Control connections
    - Mask programming
    - Fuse
    - Antifuse
    - Single-bit storage element

Programming Technologies

- The technologies (continued)
  - Build lookup tables
    - Storage elements (as in a memory)
  - Transistor Switching Control
    - Stored charge on a floating transistor gate
      - Erasable
      - Electrically erasable
      - Flash (as in Flash Memory)
    - Storage elements (as in a memory)

Technology Characteristics

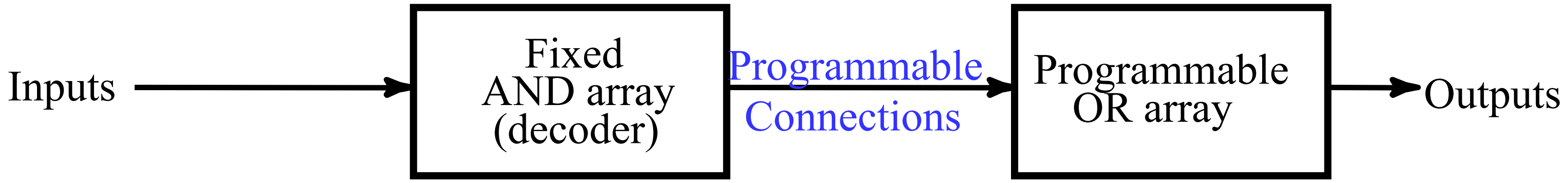
- **Permanent - Cannot be erased and reprogrammed**
    - **Mask programming**
    - **Fuse**
    - **Antifuse**
- **Reprogrammable**
    - **Volatile - Programming lost if chip power lost**
        - **Single-bit storage element**
    - **Non-Volatile**
        - **Erasable**
        - **Electrically erasable**
        - **Flash (as in Flash Memory)**

Programmable Configurations

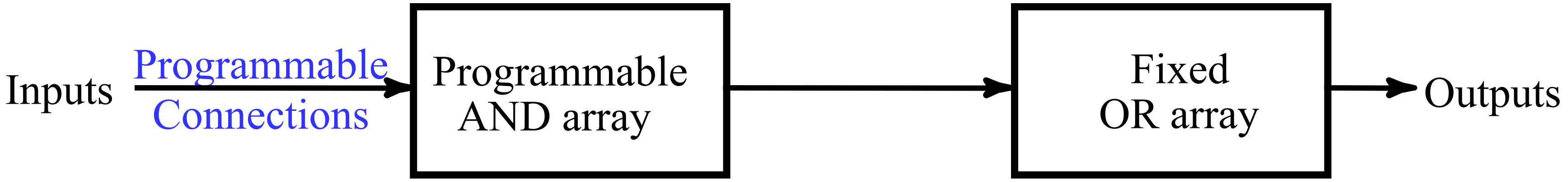
- *Read Only Memory* (*ROM*) - a fixed array of AND gates and a programmable array of OR gates
- *Programmable Array Logic* (*PAL*)® - a programmable array of AND gates feeding a fixed array of OR gates.
- *Programmable Logic Array* (*PLA*) - a programmable array of AND gates feeding a programmable array of OR gates.
- *Complex Programmable Logic Device* (*CPLD*) /*Field- Programmable Gate Array* (*FPGA*) - complex enough to be called "architectures" - See VLSI Programmable Logic Devices reading supplement

PAL is a registered trademark of Lattice Semiconductor Corp.

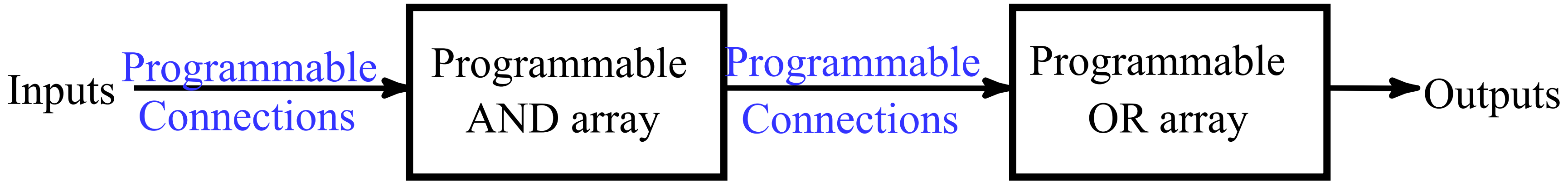
ROM, PAL and PLA Configurations



(a) Programmable read-only memory (PROM)



(b) Programmable array logic (PAL) device



(c) Programmable logic array (PLA) device

Read Only Memory

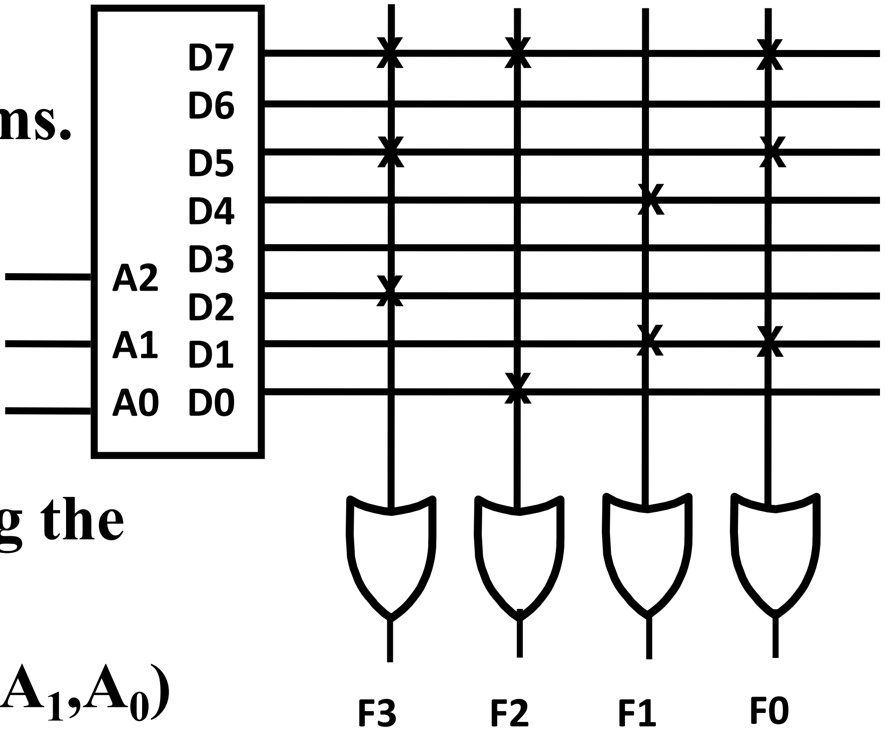
- **Read Only Memories (ROM) or Programmable Read Only Memories (PROM) have:**
  - **N input lines,**
  - **M output lines, and**
  - **2^N decoded minterms.**
- **Fixed AND array with 2^N outputs implementing all N-literal minterms.**
- **Programmable OR Array with M outputs lines to form up to M sum of minterm expressions.**

Read Only Memory

- **A program for a ROM or PROM is simply a multiple-output truth table**
  - **If a 1 entry, a connection is made to the corresponding minterm for the corresponding output**
  - **If a 0, no connection is made**
- **Can be viewed as a *memory* with the inputs as *addresses* of *data* (output values), hence ROM or PROM names!**

Read Only Memory Example

- Example: A 8 X 4 ROM (N = 3 input lines, M= 4 output lines)
- The fixed "AND" array is a "decoder" with 3 inputs and 8 outputs implementing minterms.
- The programmable "OR" array uses a single line to represent all inputs to an OR gate. An "X" in the array corresponds to attaching the minterm to the OR



- Read Example: For input (A_2,A_1,A_0) = 001, output is (F_3,F_2,F_1,F_0) = 0011.
- What are functions F_3, F_2, F_1 and F_0 in terms of (A_2, A_1, A_0)?

Example: Square of 3-bit input number

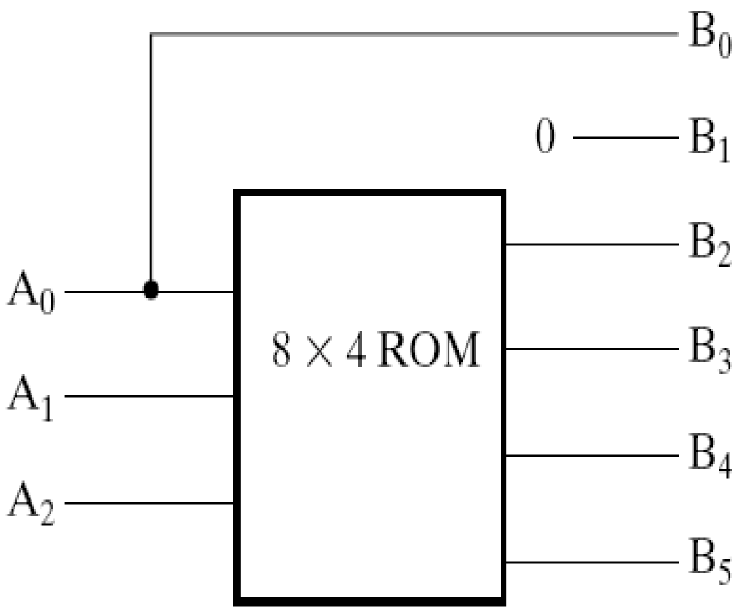
Inputs			Outputs						
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

$B_0 = A_0$

$B_1 = 0$

Example: Square of 3-bit input number

- **8 × 4 ROM are selected**



ROM Truth Table

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

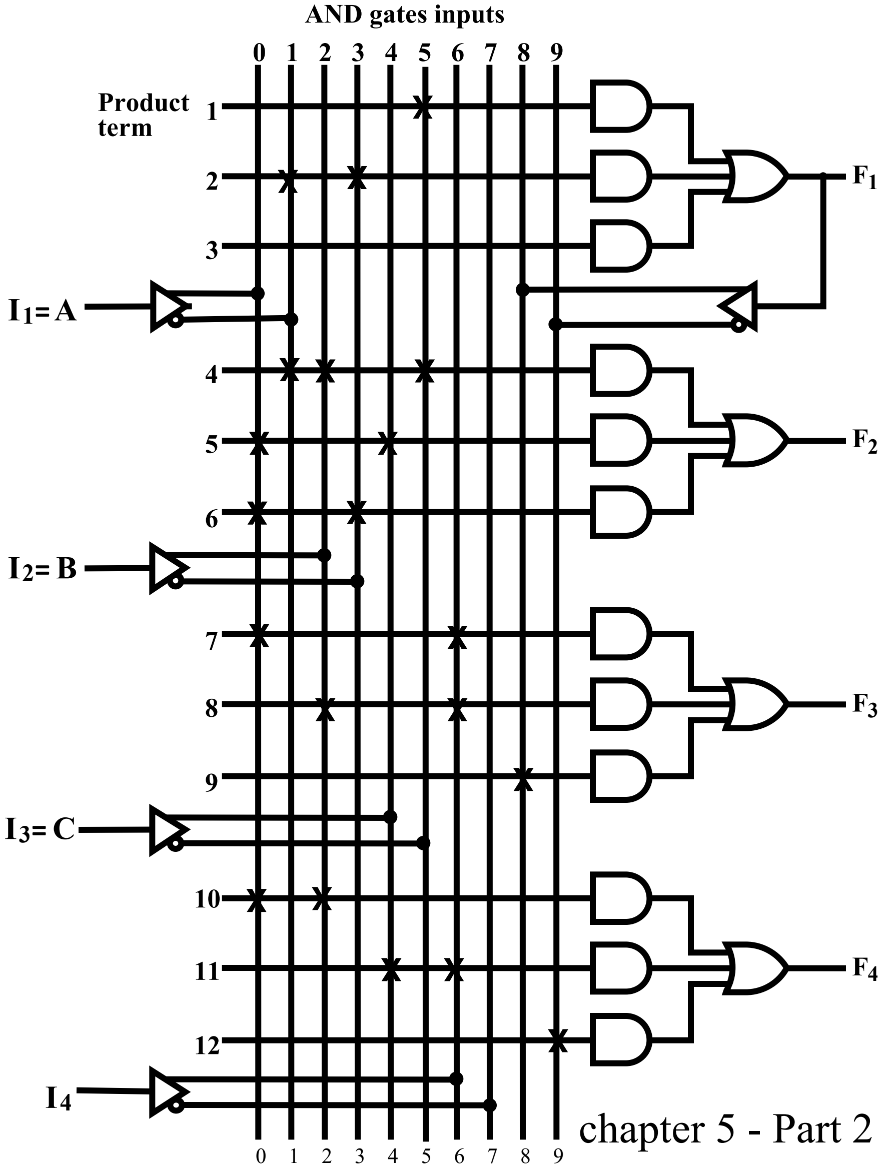
Programmable Array Logic (PAL)

- The PAL is the opposite of the ROM, having a <u>programmable</u> set of ANDs combined with <u>fixed</u> ORs.
- Disadvantage
  - ROM guaranteed to implement any M functions of N inputs. PAL may have too few inputs to the OR gates.
- Advantages
  - For given internal complexity, a PAL can have larger N and M
  - Some PALs have outputs that can be complemented, adding POS functions
  - No multilevel circuit implementations in ROM (without external connections from output to input). PAL has outputs from OR terms as internal inputs to all AND terms, making implementation of multi-level circuits easier.

Programmable Array Logic Example

- **4-input, 4-output PAL with fixed, 3-input OR terms**
- **What are the equations for F1 through F4?**

$$F1 = \overline{A}\,\overline{B} + \overline{C}$$
$$F2 = \overline{A}B\overline{C} + AC + A\overline{B}$$
$$F3 =$$
$$F4 =$$



Programmable Array Logic

- Design requires fitting functions within the limited number of ANDs per OR gate
- Single function optimization is the first step to fitting
- Otherwise, if the number of terms in a function is greater than the number of ANDs per OR gate, then factoring is necessary

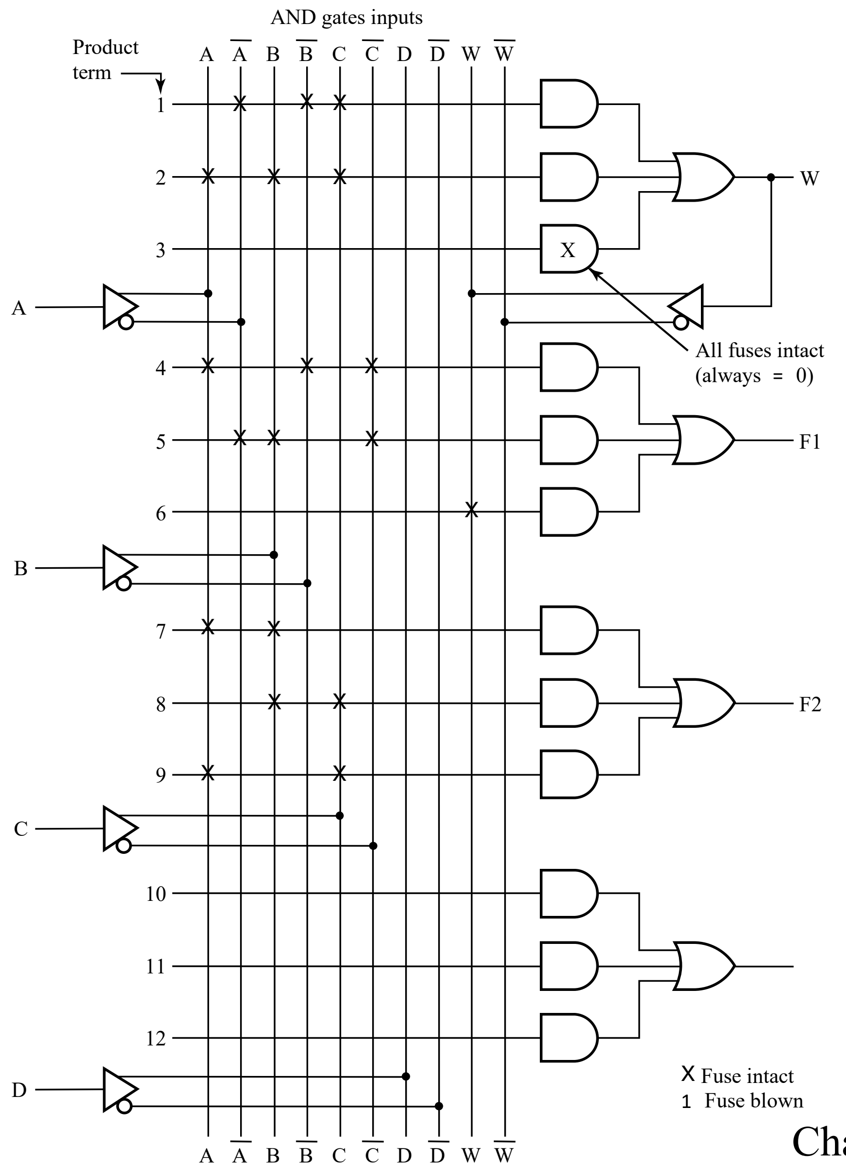
Programmable Array Logic Example

- Equations: $F1 = A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + \overline{A}\,\overline{B}C + ABC$

  $F2 = AB + BC + AC$
- F1 must be factored since four terms
- Factor out last two terms as W

Product term	AND Inputs A	B	C	D	W	Outputs
1	0	0	1	—	—	$W = \overline{A}\,\overline{B}C + ABC$
2	1	1	1	—	—	
3	—	—	—	—	—	
4	1	0	0	—	—	$F1 = X =$ $A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + W$
5	0	1	0	—	—	
6	—	—	—	—	1	
7	1	1	—	—	—	$F2 = Y =$ $AB + BC + AC$
8	—	1	1	—	—	
9	1	—	1	—	—	
10	—	—	—	—	—	
11	—	—	—	—	—	
12	—	—	—	—	—	

Programmable Array Logic Example


AND gates inputs
Product term
A $\overline{A}$ B $\overline{B}$ C $\overline{C}$ D $\overline{D}$ W $\overline{W}$
1
2
3
X
W
A
All fuses intact (always = 0)
4
5
6
F1
B
7
8
9
F2
C
10
11
12
D
X Fuse intact
1 Fuse blown

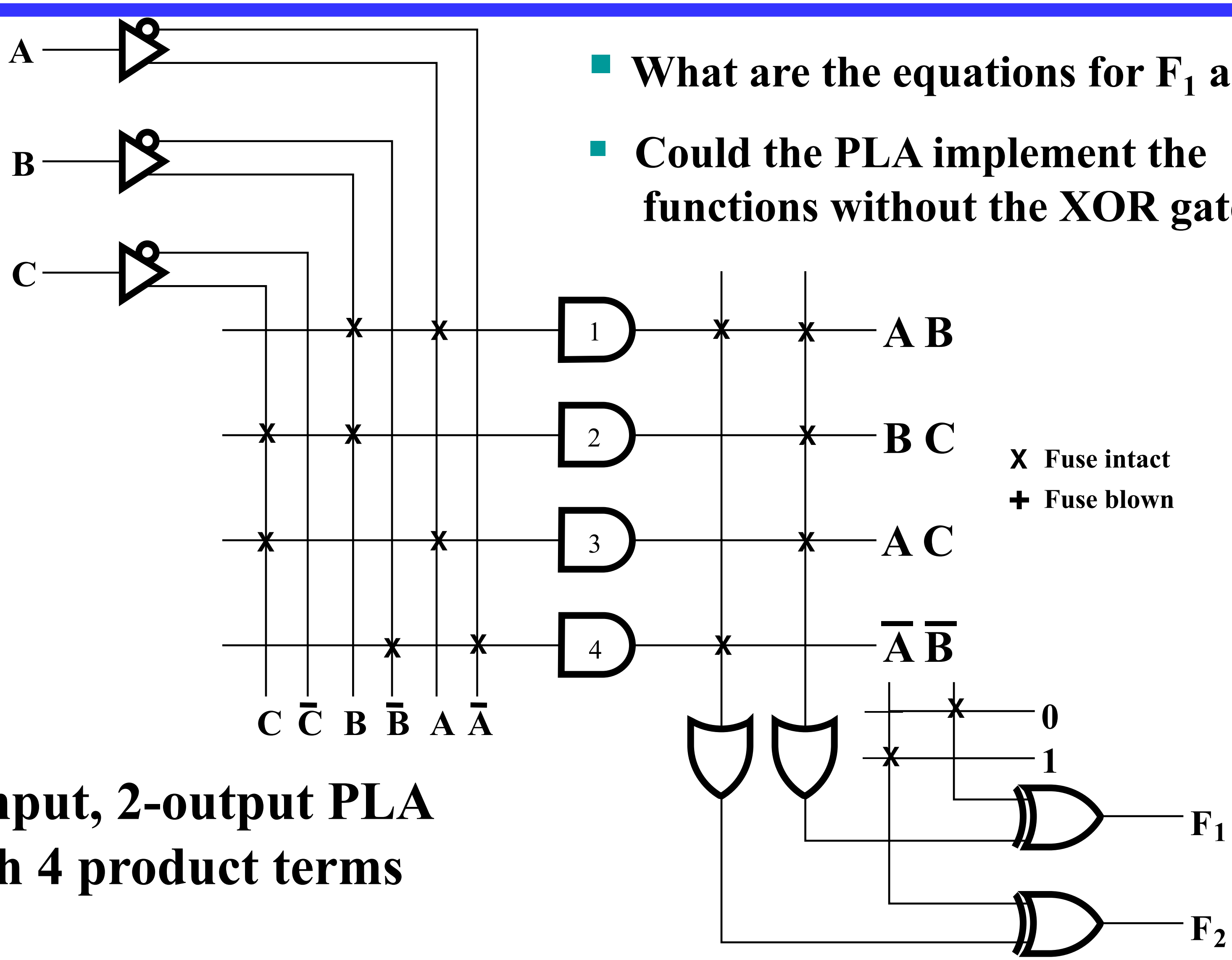
Programmable Logic Array (PLA)

- Compared to a ROM and a PAL, a PLA is the most flexible having a <u>programmable</u> set of ANDs combined with a <u>programmable</u> set of ORs.
- Advantages
  - A PLA can have large N and M permitting implementation of equations that are impractical for a ROM (because of the number of inputs, N, required)
  - A PLA has all of its product terms connectable to all outputs, overcoming the problem of the limited inputs to the PAL Ors
  - Some PLAs have outputs that can be complemented, adding POS functions

Programmable Logic Array (PLA)

- **Disadvantages**
  - Often, the product term count limits the application of a PLA.
  - Two-level multiple-output optimization is required to reduce the number of product terms in an implementation, helping to fit it into a PLA.
  - Multi-level circuit capability available in PAL not available in PLA. PLA requires external connections to do multi-level circuits.

Programmable Logic Array Example



- What are the equations for F_1 and F_2?
- Could the PLA implement the functions without the XOR gates?

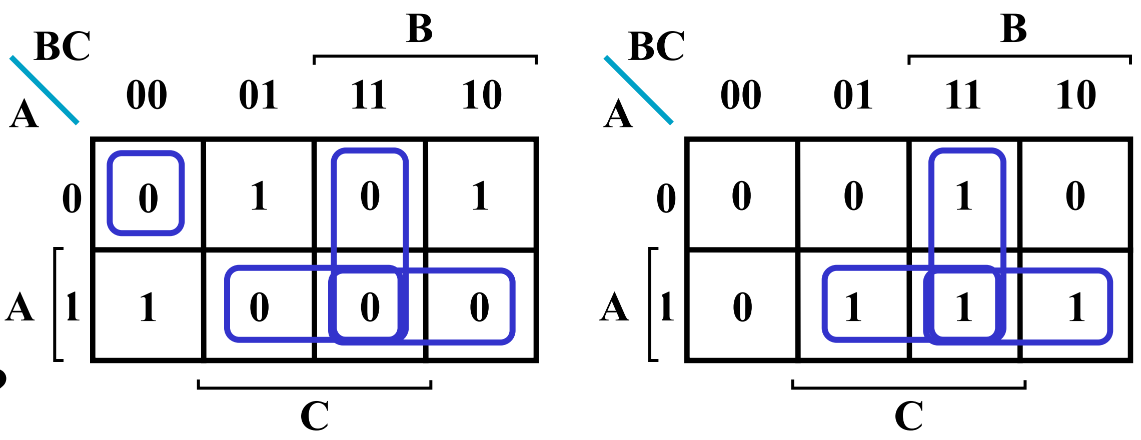
- 3-input, 2-output PLA with 4 product terms

Programmable Logic Array

- The set of functions to be implemented must fit the available number of product terms
- The number of literals per term is less important in fitting
- The best approach to fitting is multiple-output, two-level optimization (which has not been discussed)
- Since output inversion is available, terms can implement either a function or its complement
- For small circuits, K-maps can be used to visualize product term sharing and use of complements
- For larger circuits, software is used to do the optimization including use of complemented functions

Programmable Logic Array Example

- K-map specification
- How can this be implemented with four terms?
- Complete the programming table



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

$\overline{F}_1 = AB + AC + BC + \overline{A}\,\overline{B}\,\overline{C}$

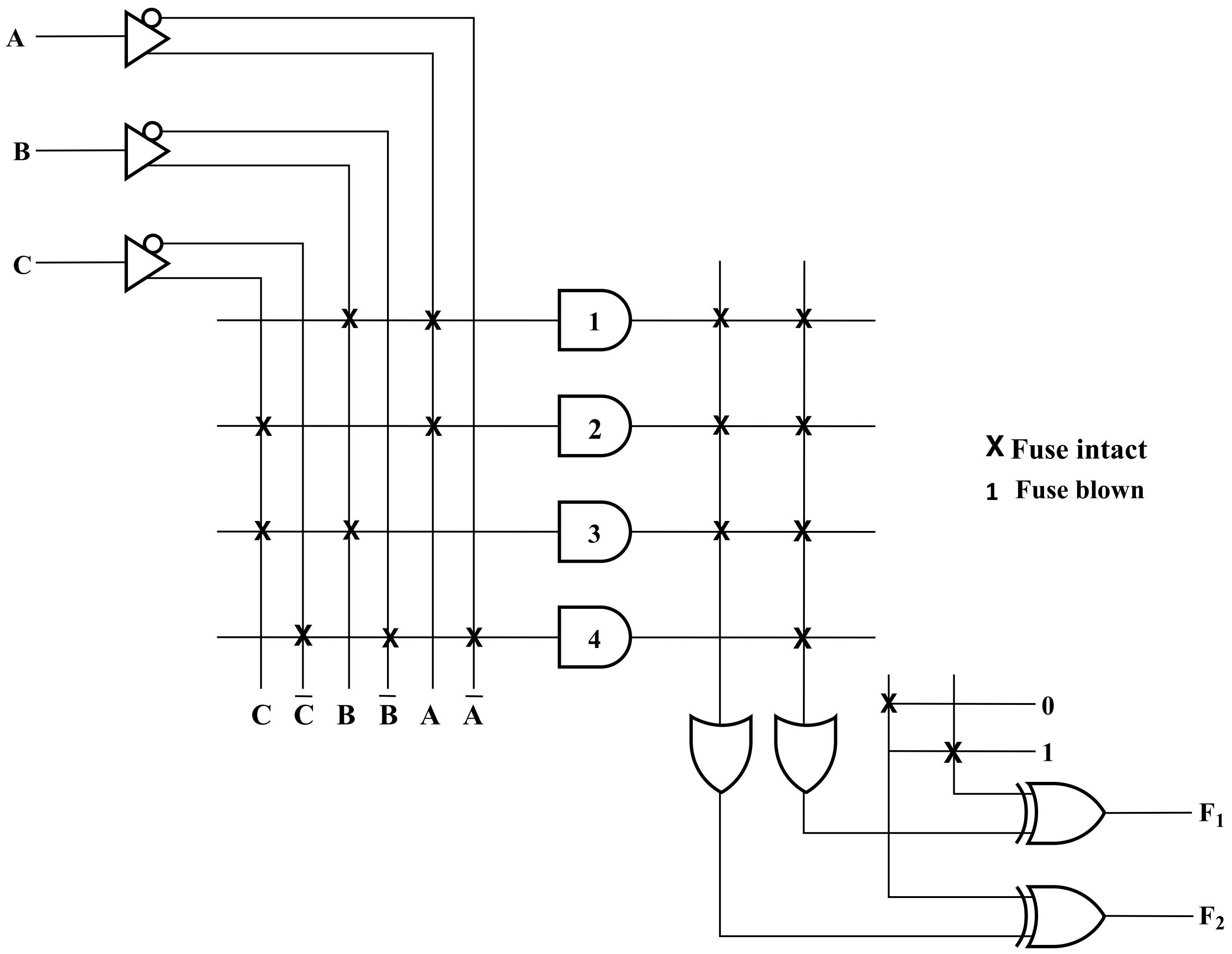
$F_2 = AB + AC + BC$

$\overline{F}_2 = \overline{A}\,\overline{C} + \overline{A}\,\overline{B} + \overline{B}\,\overline{C}$

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	1
$\overline{A}\,\overline{B}\,\overline{C}$	4	0	0	0	1	–

Programmable Logic Array Example



X Fuse intact

1 Fuse blown

Assignments

- 5-3, 5-4, 5-12