Programmable Logic

PROGRAMMABLE LOGIC DEVICES (PLDs)

PROM Programmable Read-Only Memory
- Fixed "AND" array with 2 ^(# Inputs) minterms
- Programmable "OR" array with m output bits

EPROM Erasable PROM
- Erase with ultraviolet light, and program again

EEPROM Electrically Erasable PROM
- Erase addressed byte with 20 volts

Flash Memory High density, low cost EEPROM
- Non-selective complete erase with 20 volts

PAL Programmable Array Logic
- Programmable "AND" array with *n* product terms
- Fixed "OR" array, each of *m* outputs has *n/m* terms

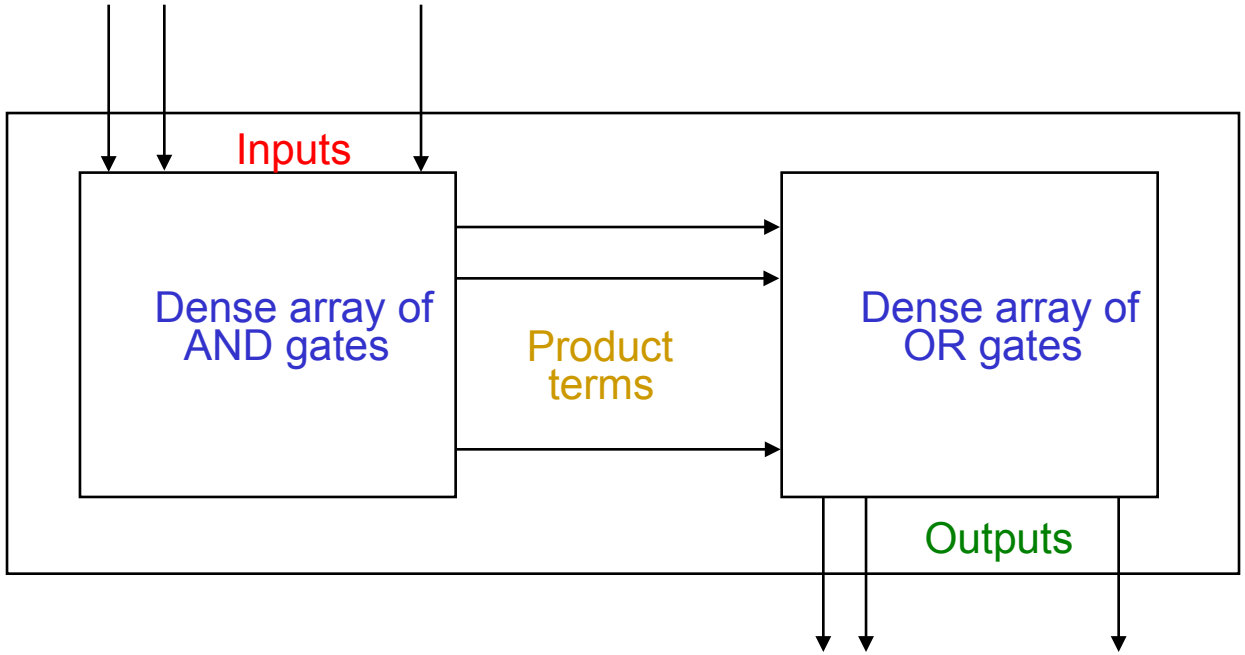
PLA Programmable Logic Array
- Programmable "AND" array with *n* product terms
- Programmable "OR" array with *m* output bits

PLS Programmable Logic Sequencer
- PLA plus Flip-Flops and extra combinational logic

FPGA Field Programmable Gate Array
- Dynamically configurable logic and I/O blocks

Prgrammable Logic Organization

- **Pre-fabricated building block of many AND/OR gates (or NOR, NAND)**
- **"Personalized" by making or breaking connections among the gates**

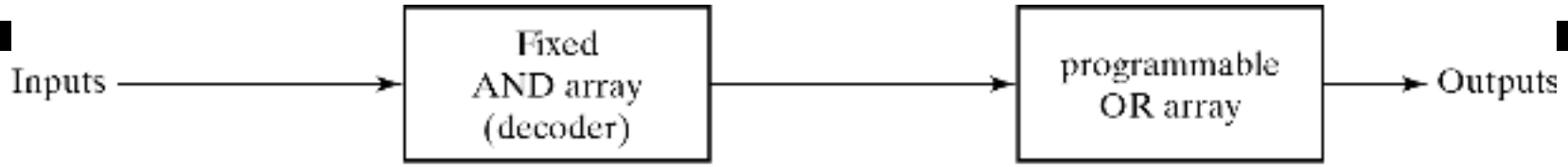


Programmable Array Block Diagram for Sum of Products Form

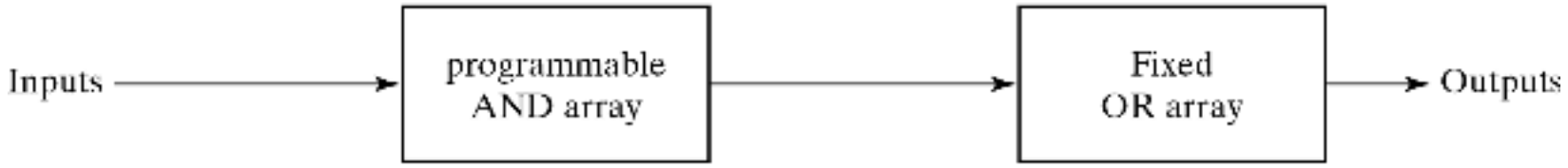
Basic Programmable Logic Organizations

- Depending on which of the AND/OR logic arrays is programmable, we have three basic organizations

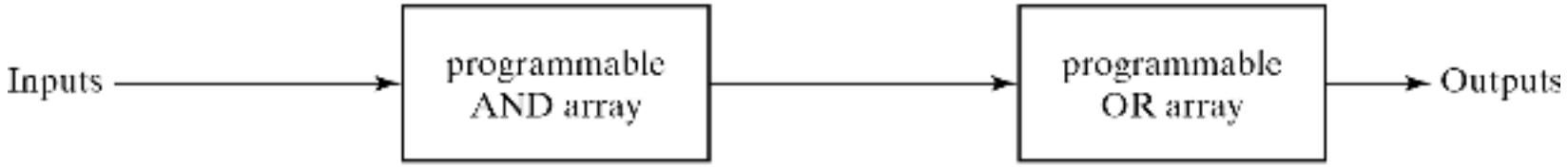
ORGANIZATION	AND ARRAY	OR ARRAY
PAL	PROG.	FIXED
PROM	FIXED	PROG.
PLA	PROG.	PROG.



(a) Programmable read-only memory (PROM)



(b) Programmable array logic (PAL)



(c) Programmable logic array (PLA)

Fig. 7-13 Basic Configuration of Three PLDs

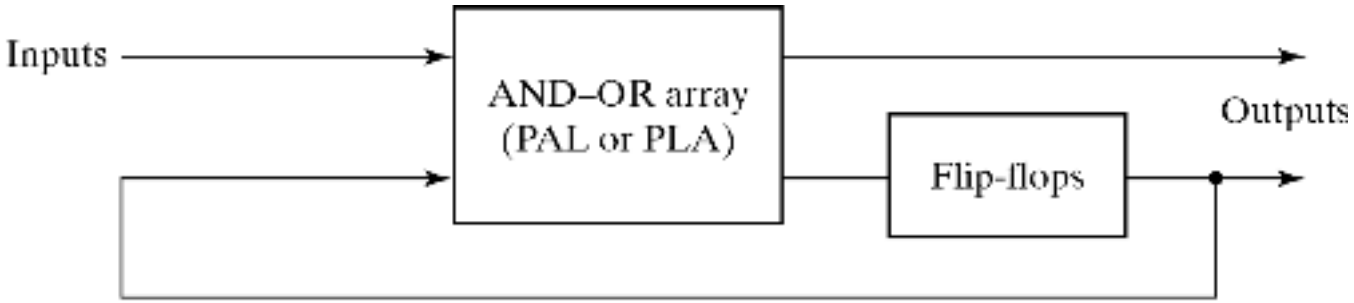


Fig. 7-18 Sequential Programmable Logic Device

PLA Logic Implementation

Key to Success: Shared Product Terms

Equations

Example:

$F0 = A + \overline{B}\,\overline{C}$
$F1 = A\,\overline{C} + A\,B$
$F2 = \overline{B}\,\overline{C} + A\,B$
$F3 = \overline{B}\,C + A$

Personality Matrix

Product term	Inputs A	B	C	Outputs F_0	F_1	F_2	F_3
A B	1	1	-	0	(1)	(1)	0
$\overline{B}$ C	-	0	1	0	0	0	(1)
A $\overline{C}$	1	-	0	0	(1)	0	0
$\overline{B}$ $\overline{C}$	-	0	0	(1)	0	(1)	0
A	1	-	-	(1)	0	0	(1)

Input Side:

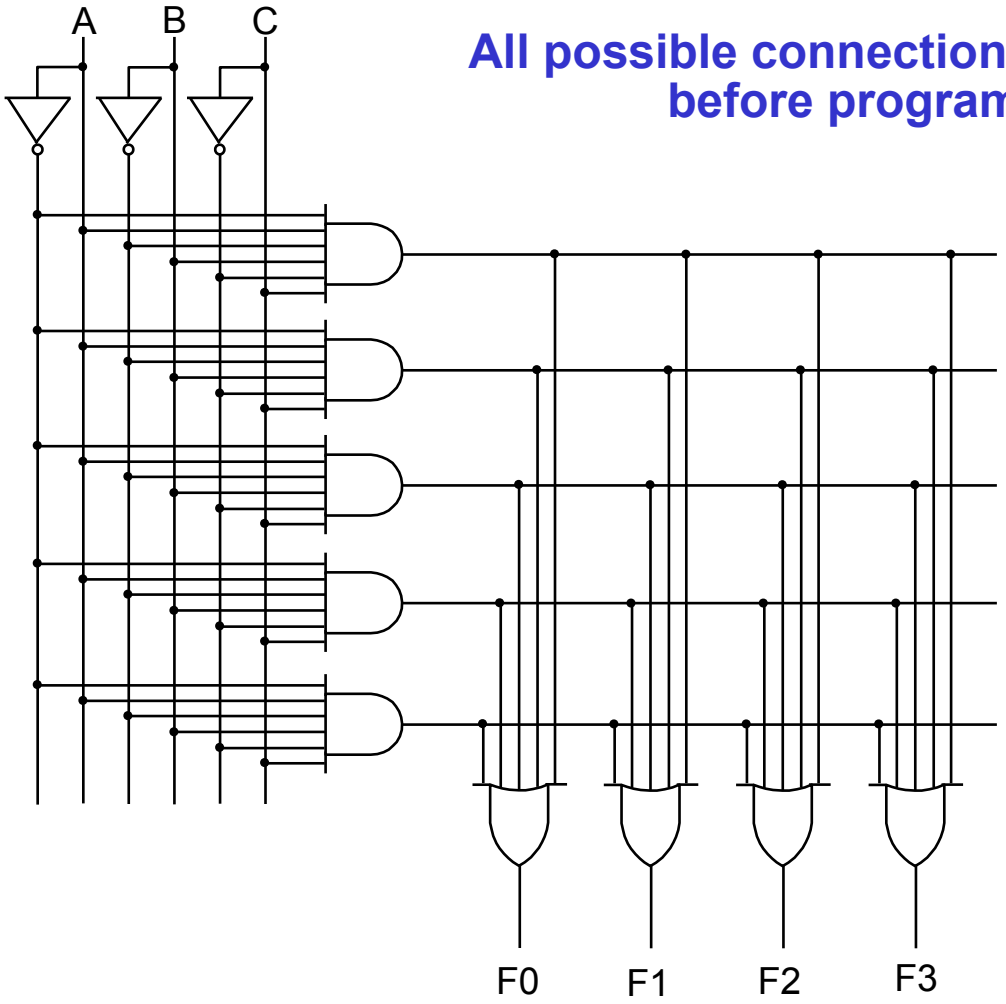
1 = asserted in term
0 = negated in term
- = does not participate

Output Side:

1 = term connected to output
0 = no connection to output

PLA Logic Implementation

Example Continued - Unprogrammed device



All possible connections are available before programming

PLA Table Generation

$F_1 = AB' + C$

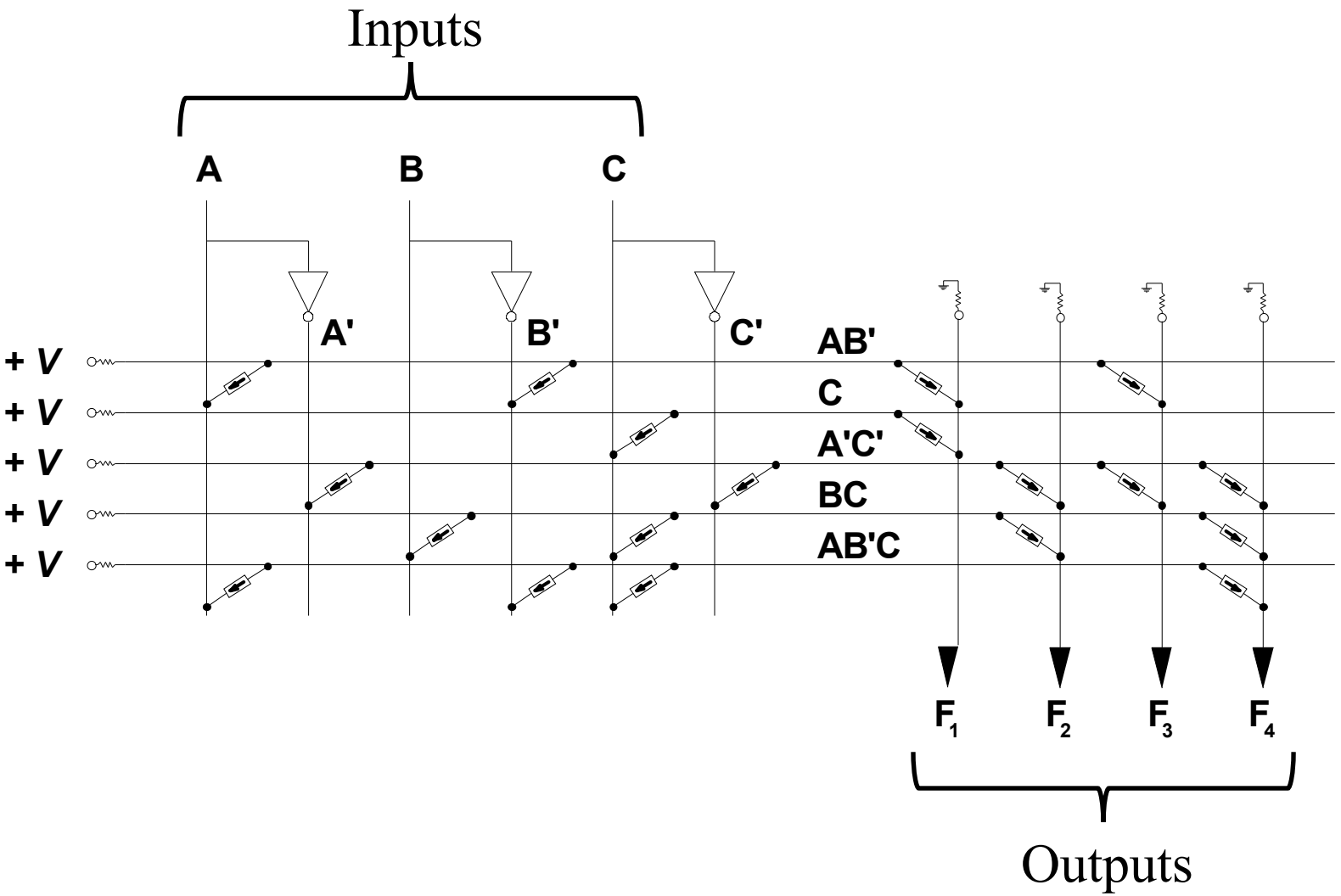
$F_2 = A'C' + BC$

$F_3 = AB' + A'C'$

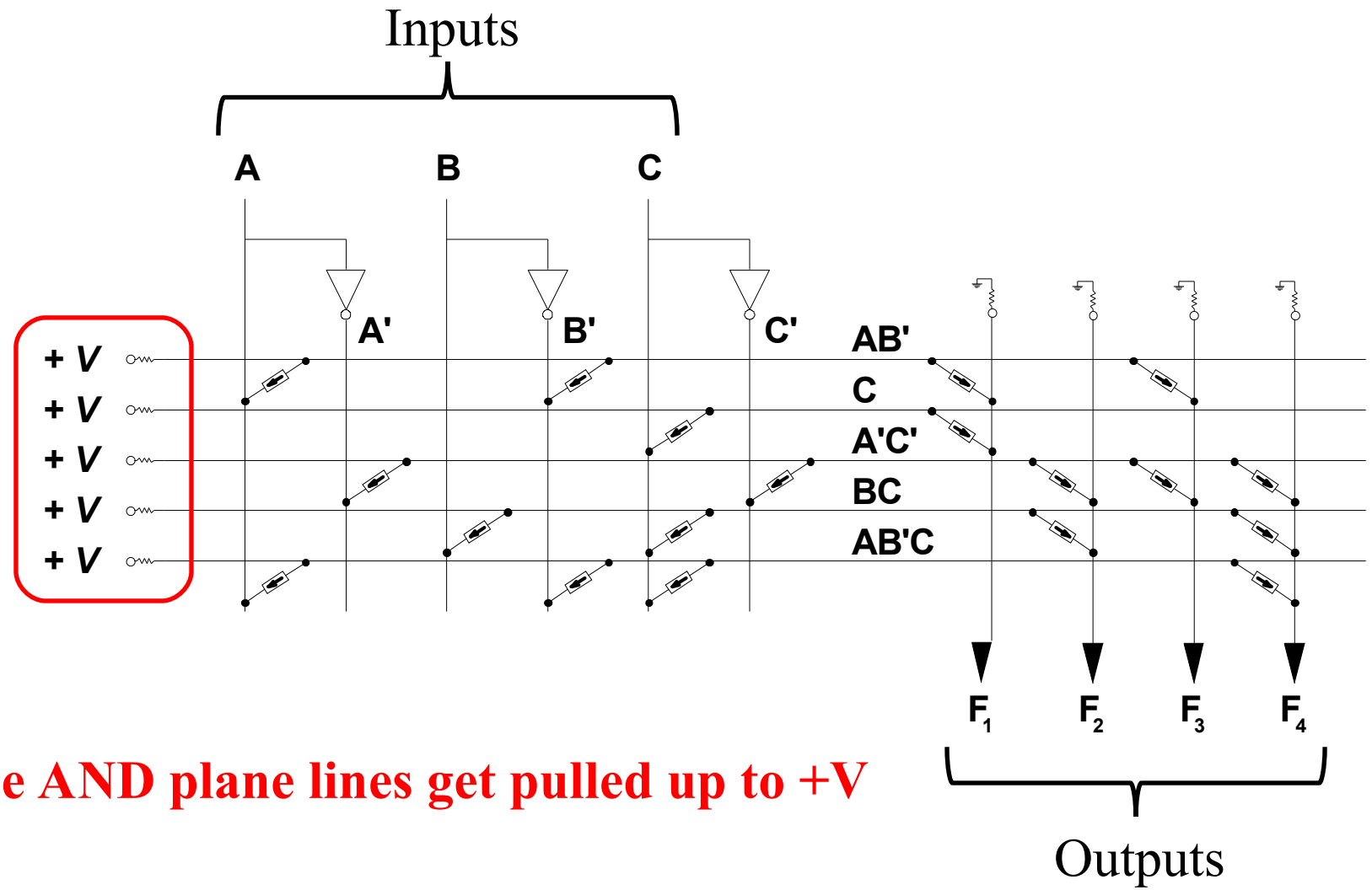
$F_4 = A'C' + BC + AB'C$

Product Term	Input (Actual) A	A'	B	B'	C	C'	Input (Specified) A	B	C	Outputs F_1	F_2	F_3	F_4
AB'	1	0	0	1	0	0	1	0	-	1	0	1	0
C	0	0	0	0	1	0	-	-	1	1	0	0	0
A'C'	0	1	0	0	0	1	0	-	0	0	1	1	1
BC	0	0	1	0	1	0	-	1	1	0	1	0	1
AB'C	1	0	0	1	1	0	1	0	1	0	0	0	1

Internal PLA Structure

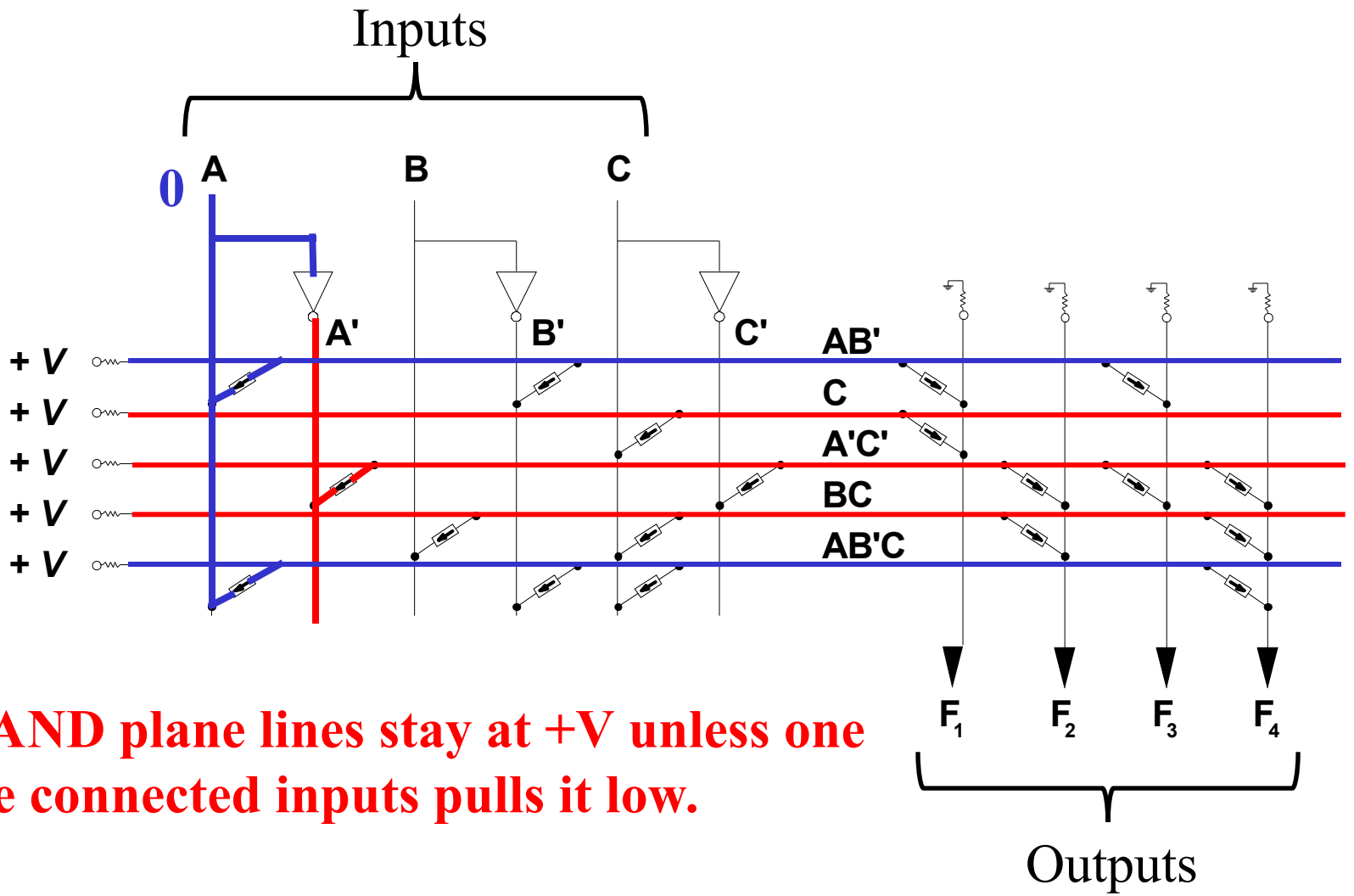


Internal PLA Structure



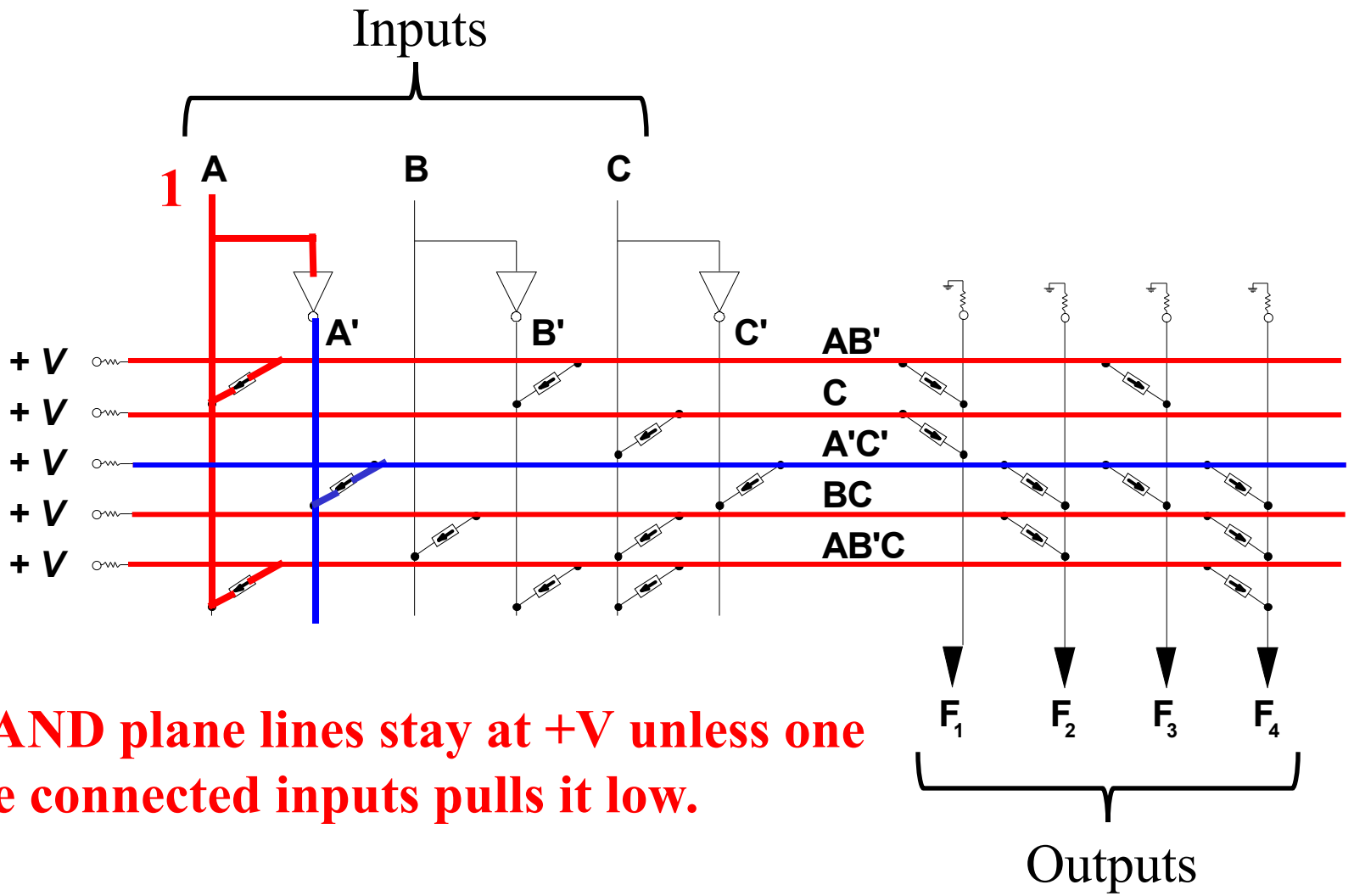
The AND plane lines get pulled up to +V

Internal PLA Structure



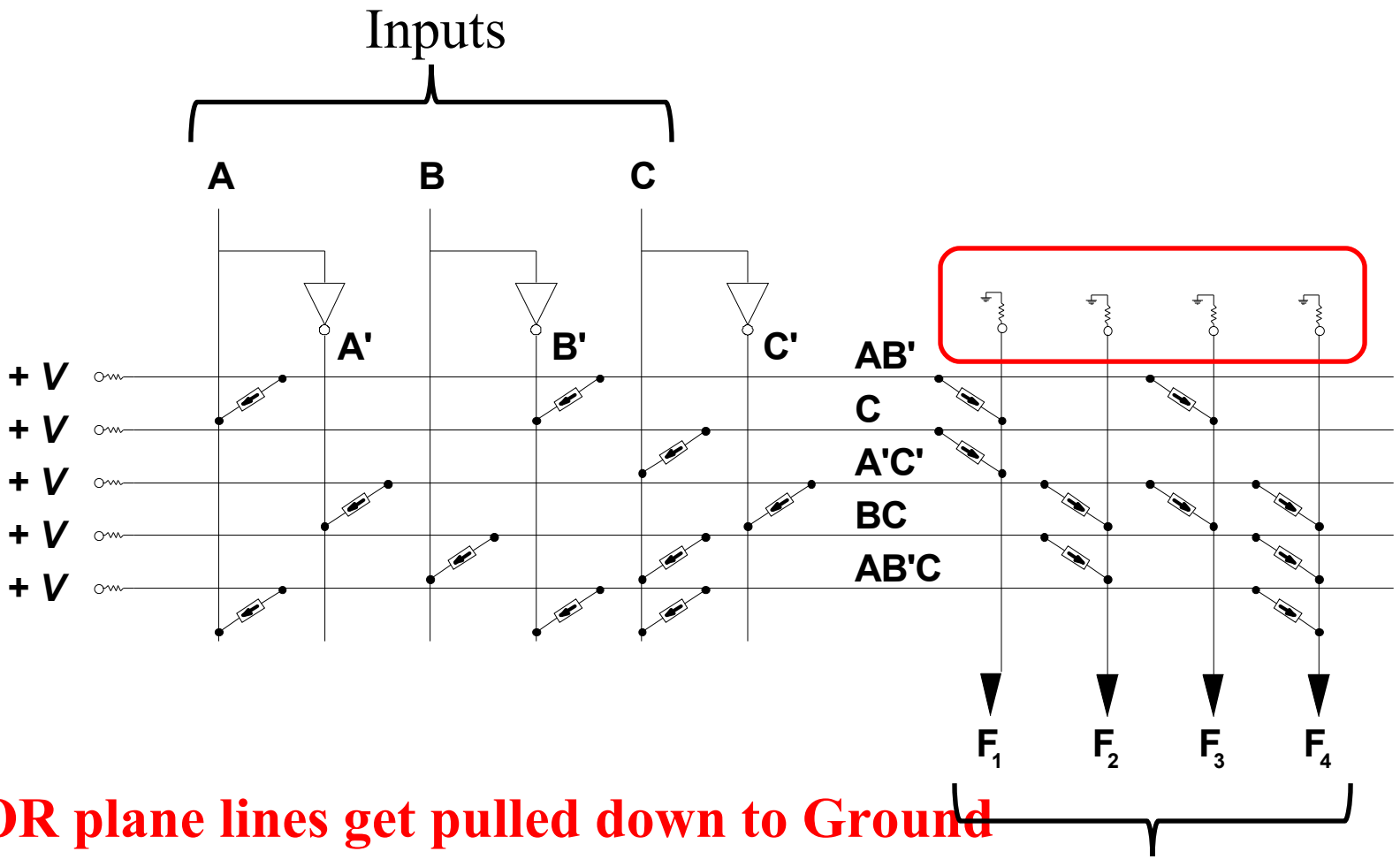
The AND plane lines stay at +V unless one of the connected inputs pulls it low.

Internal PLA Structure



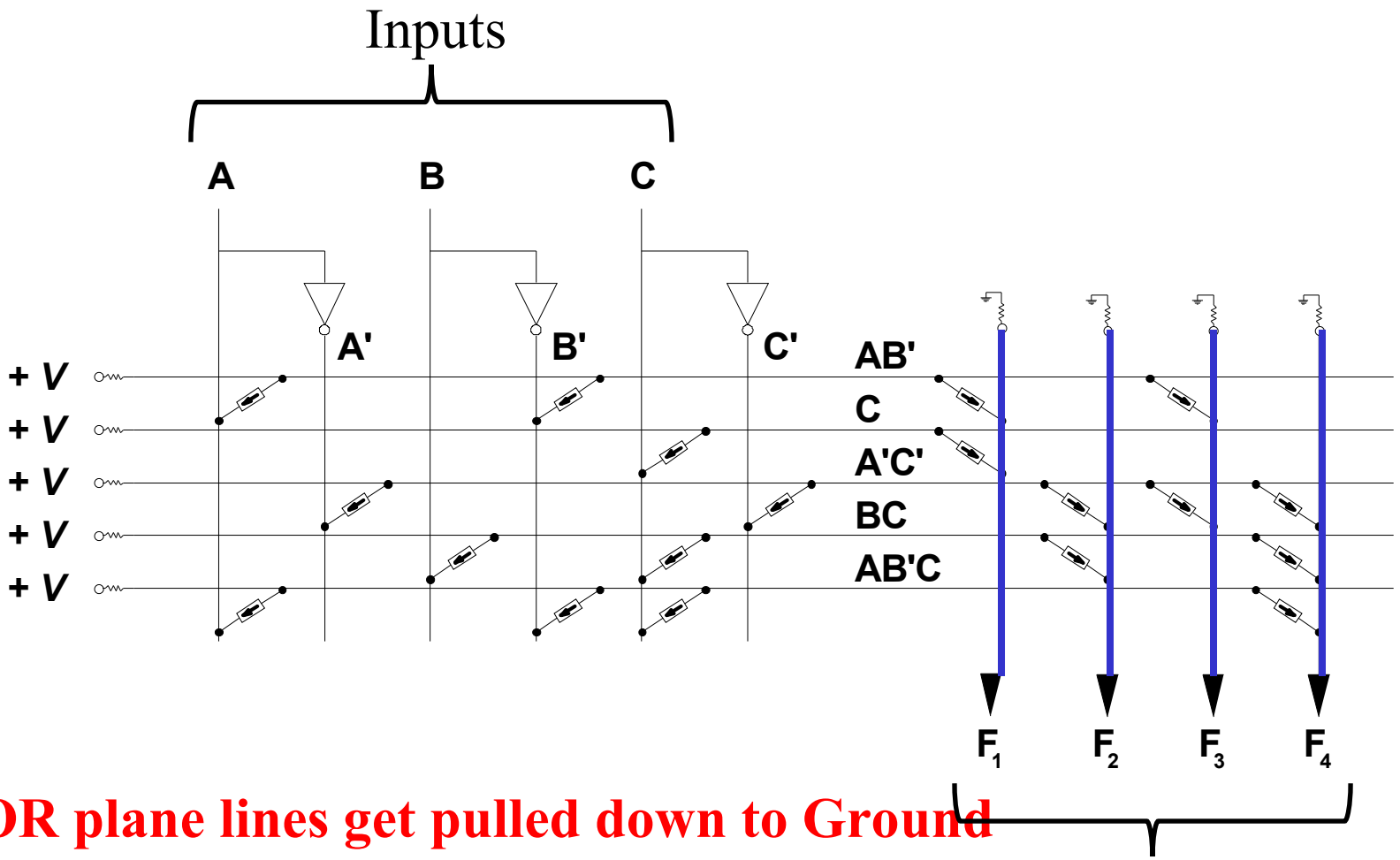
The AND plane lines stay at +V unless one of the connected inputs pulls it low.

Internal PLA Structure



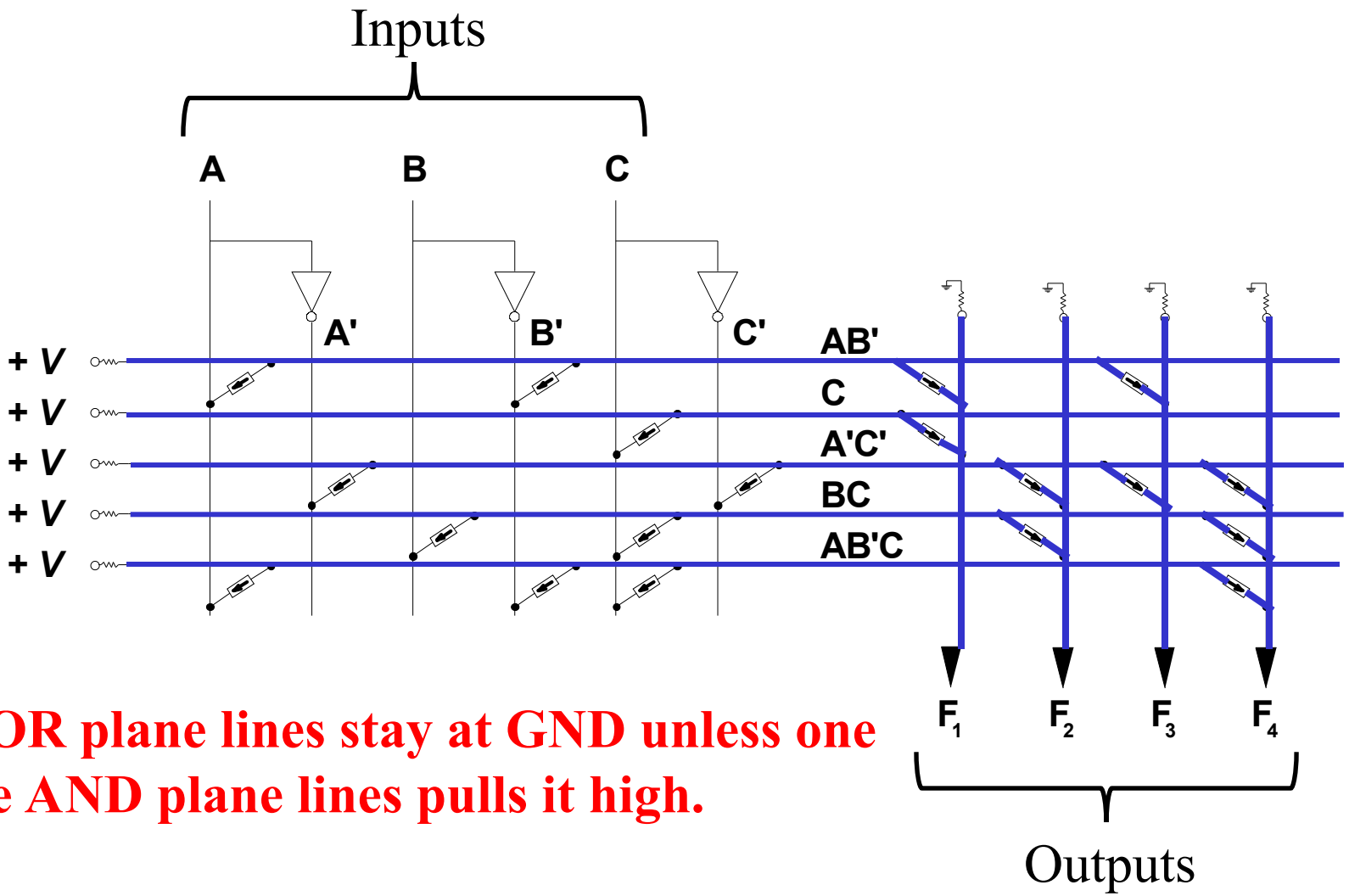
The OR plane lines get pulled down to Ground

Internal PLA Structure



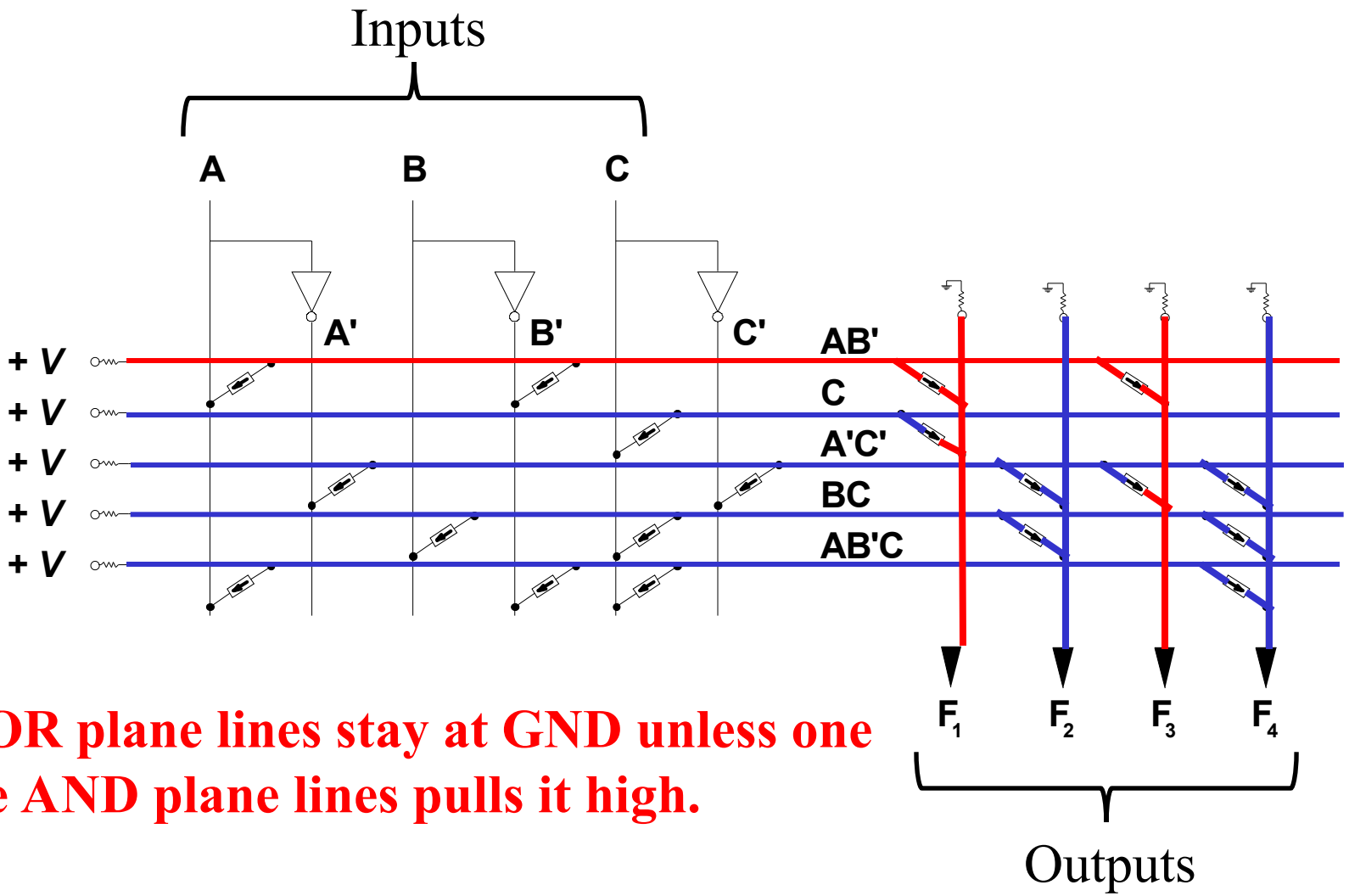
The OR plane lines get pulled down to Ground

Internal PLA Structure



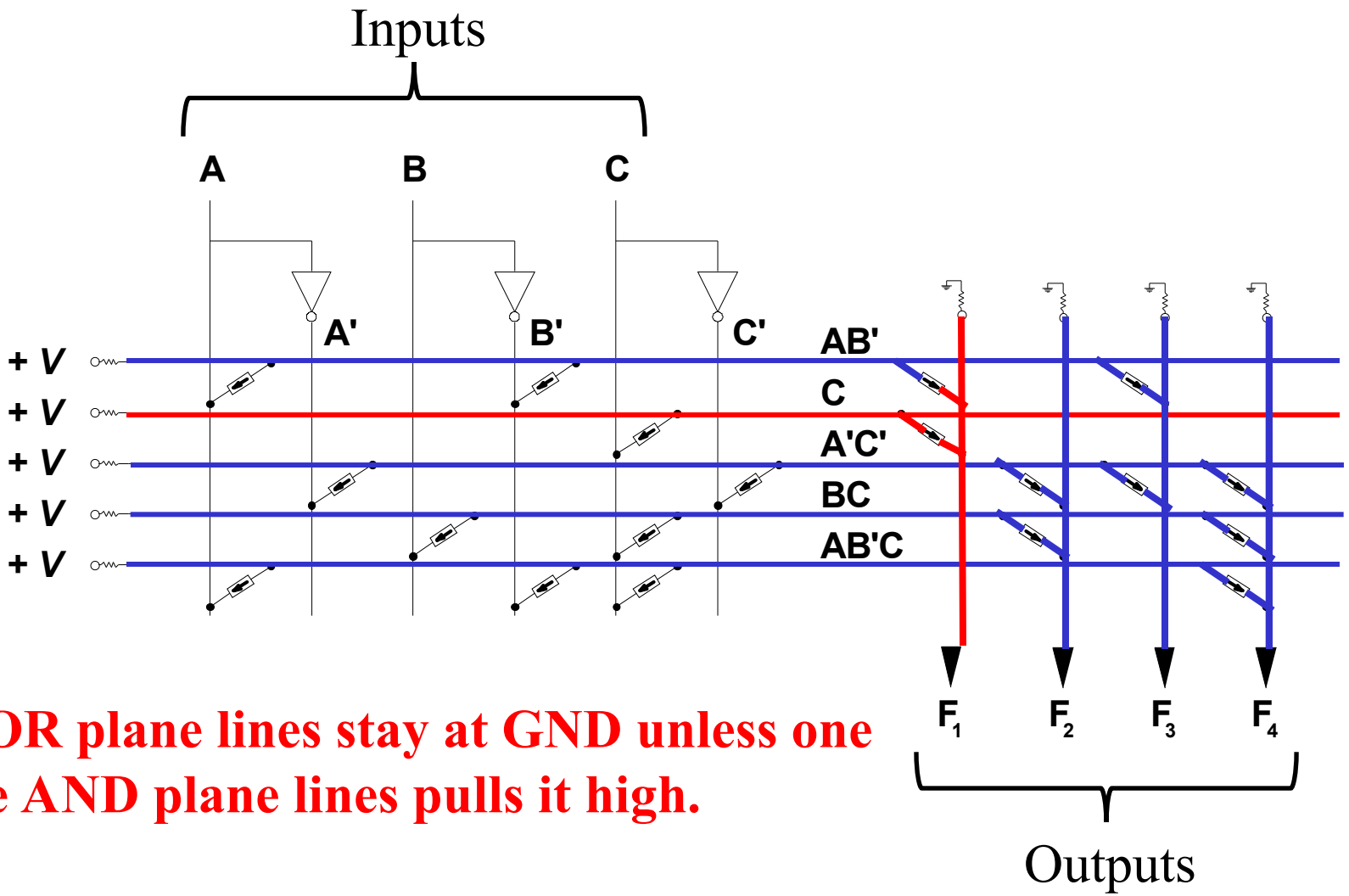
The OR plane lines stay at GND unless one of the AND plane lines pulls it high.

Internal PLA Structure



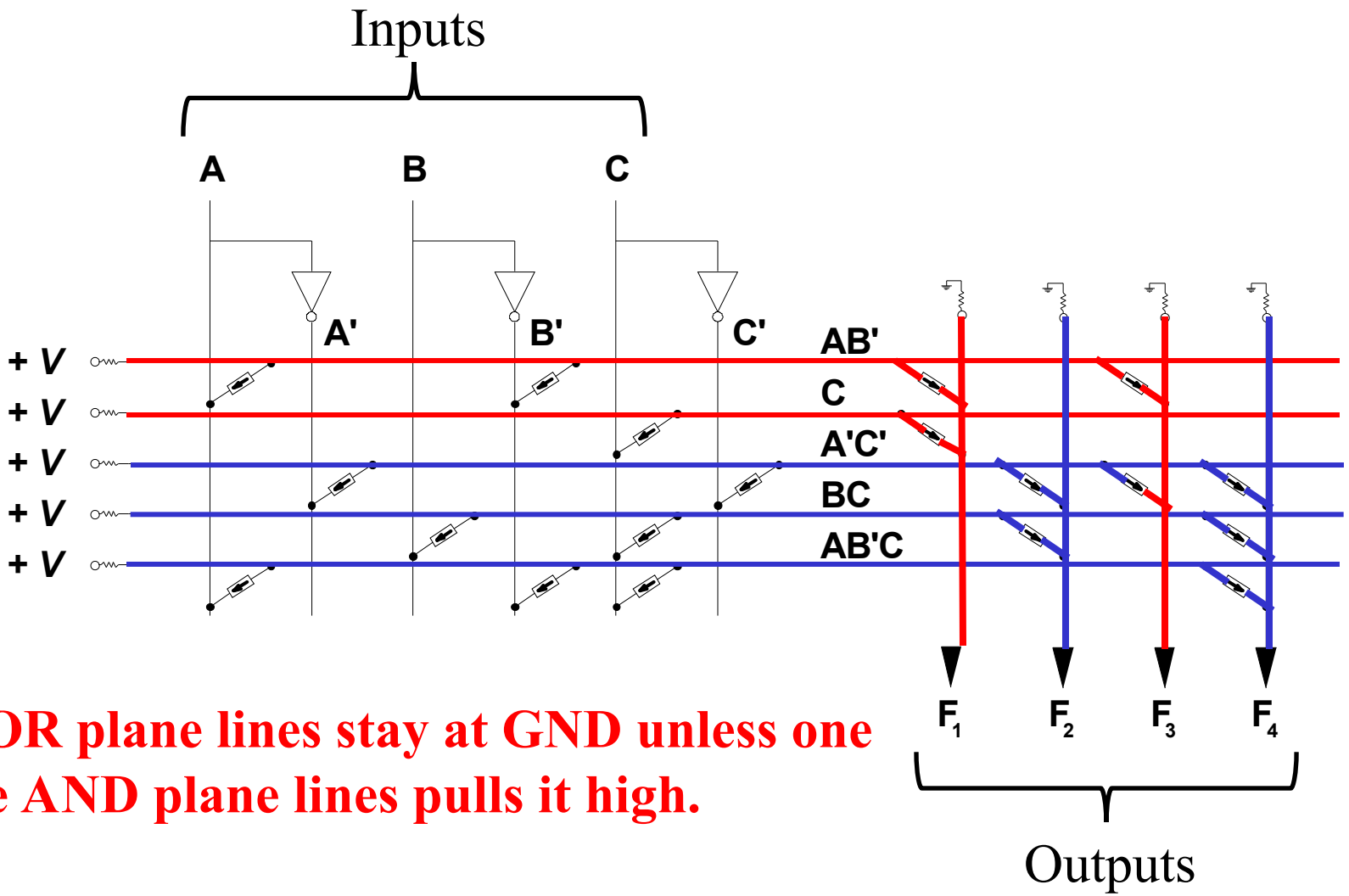
The OR plane lines stay at GND unless one of the AND plane lines pulls it high.

Internal PLA Structure



The OR plane lines stay at GND unless one of the AND plane lines pulls it high.

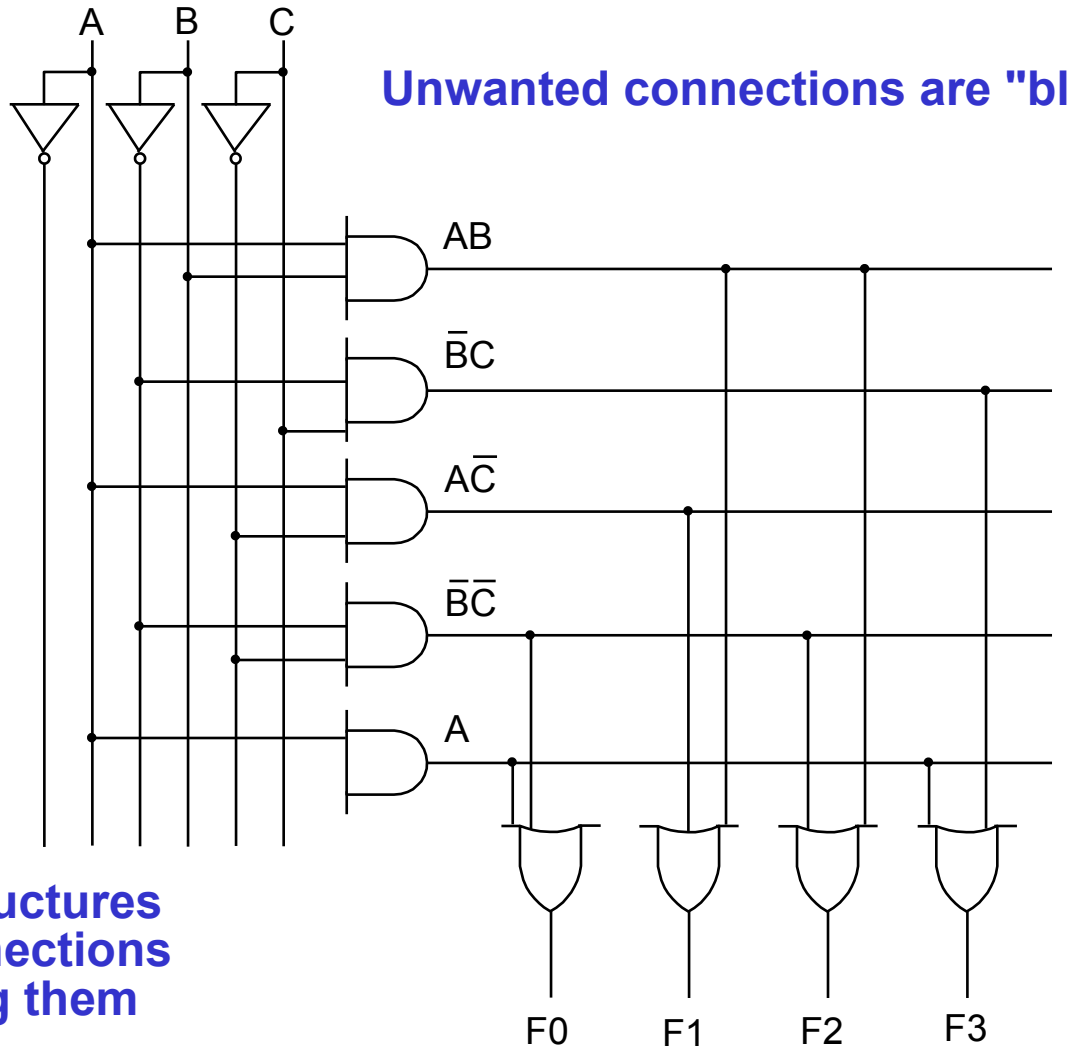
Internal PLA Structure



The OR plane lines stay at GND unless one of the AND plane lines pulls it high.

PLA Logic Implementation

Example Continued - Programmed part



Unwanted connections are "blown"

Note: some array structures work by making connections rather than breaking them

PLA Logic Implementation

Alternative representation for high fan-in structures

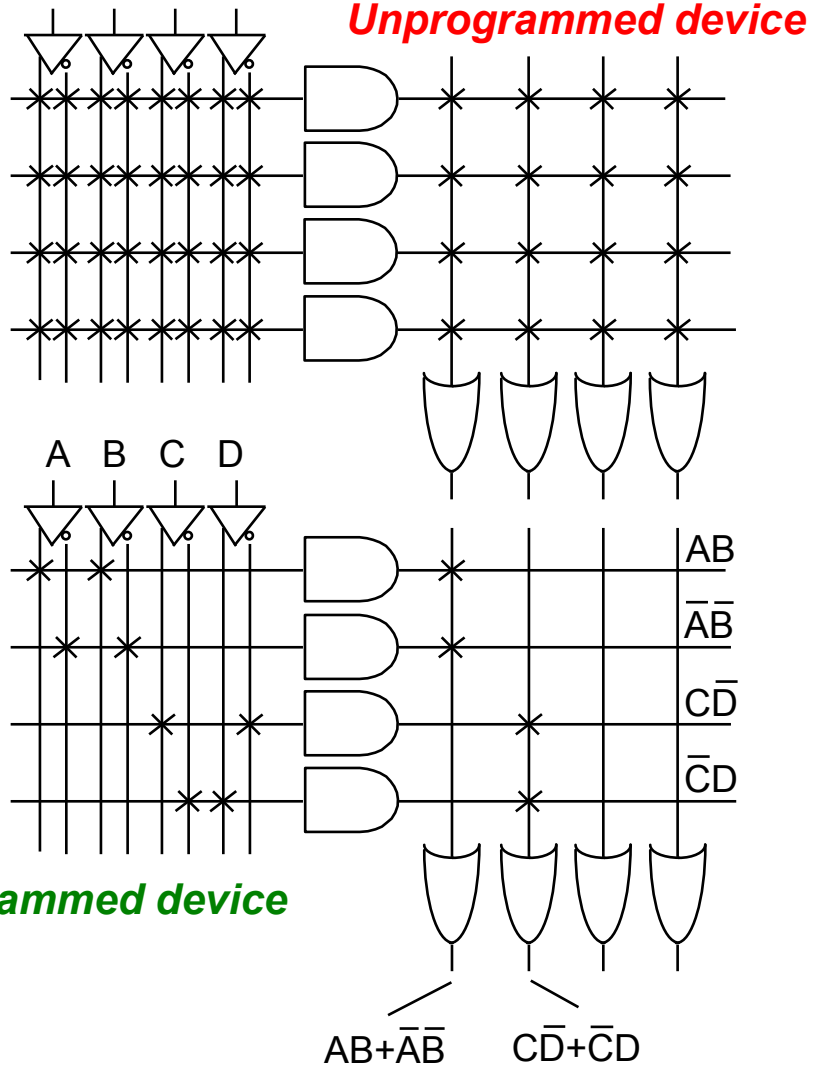
Short-hand notation so we don't have to draw all the wires!

X at junction indicates a connection

Notation for implementing

$F0 = A\,B + \overline{A}\,\overline{B}$

$F1 = C\,\overline{D} + \overline{C}\,D$



PLA Logic Implementation

Design Example

Multiple functions of A, B, C

$F1 = A\ B\ C$

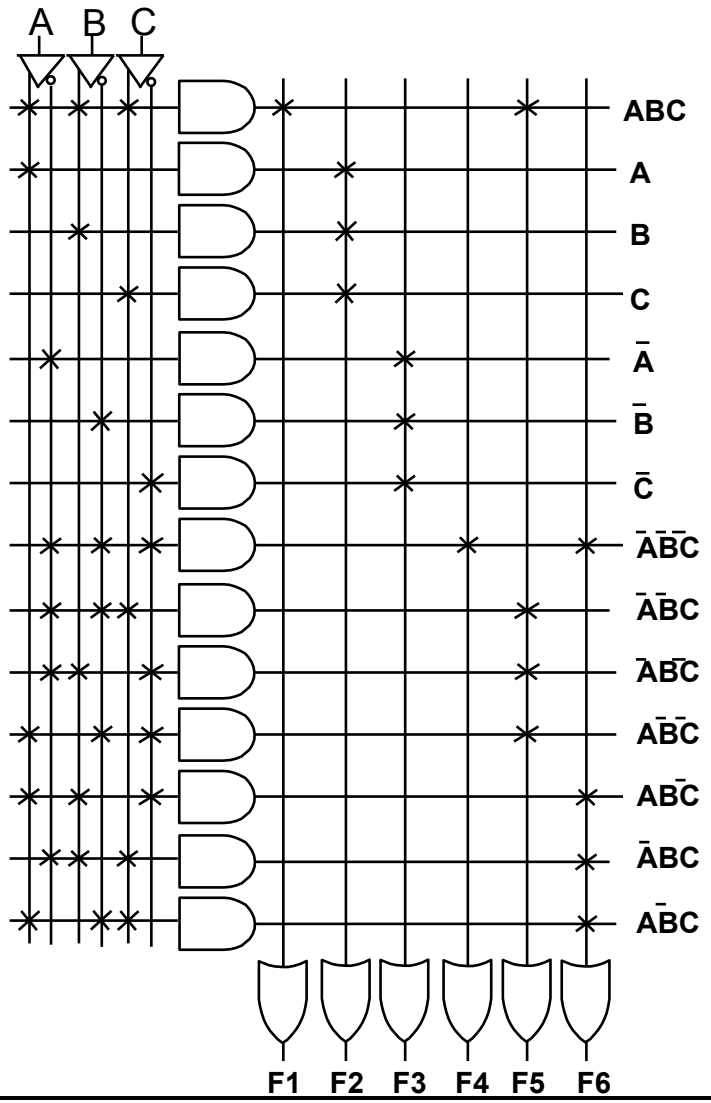
$F2 = A + B + C$

$F3 = \overline{A\ B\ C}$

$F4 = \overline{A + B + C}$

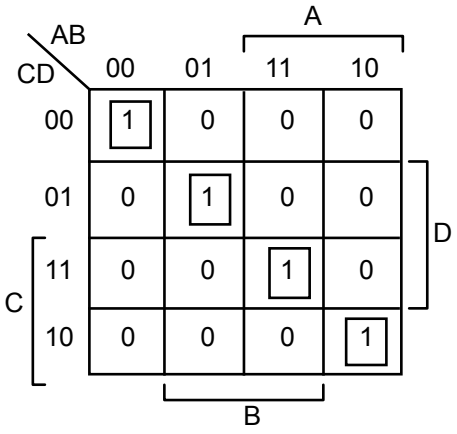
$F5 = A \oplus B \oplus C$

$F6 = \overline{A \oplus B \oplus C}$

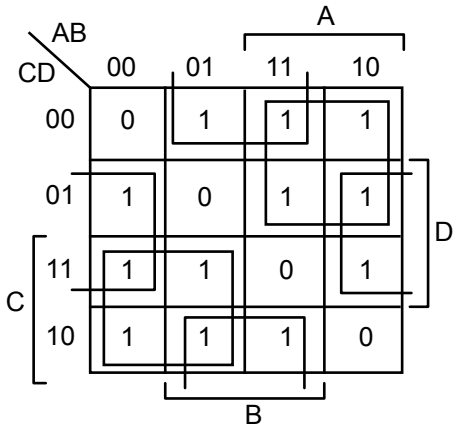


PLA Logic Implementation

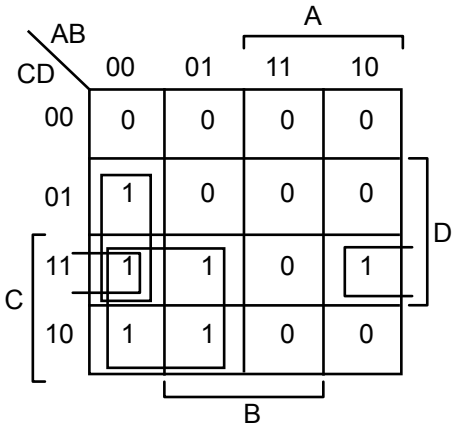
Another Example: Magnitude Comparator



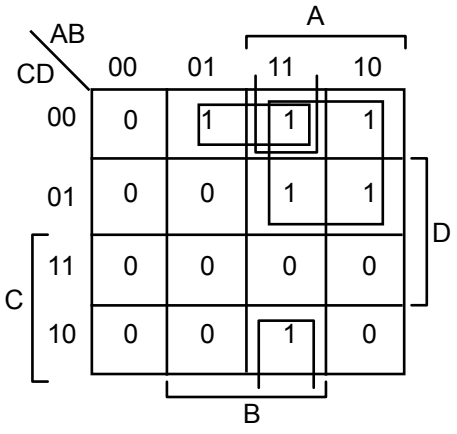
K-map for EQ



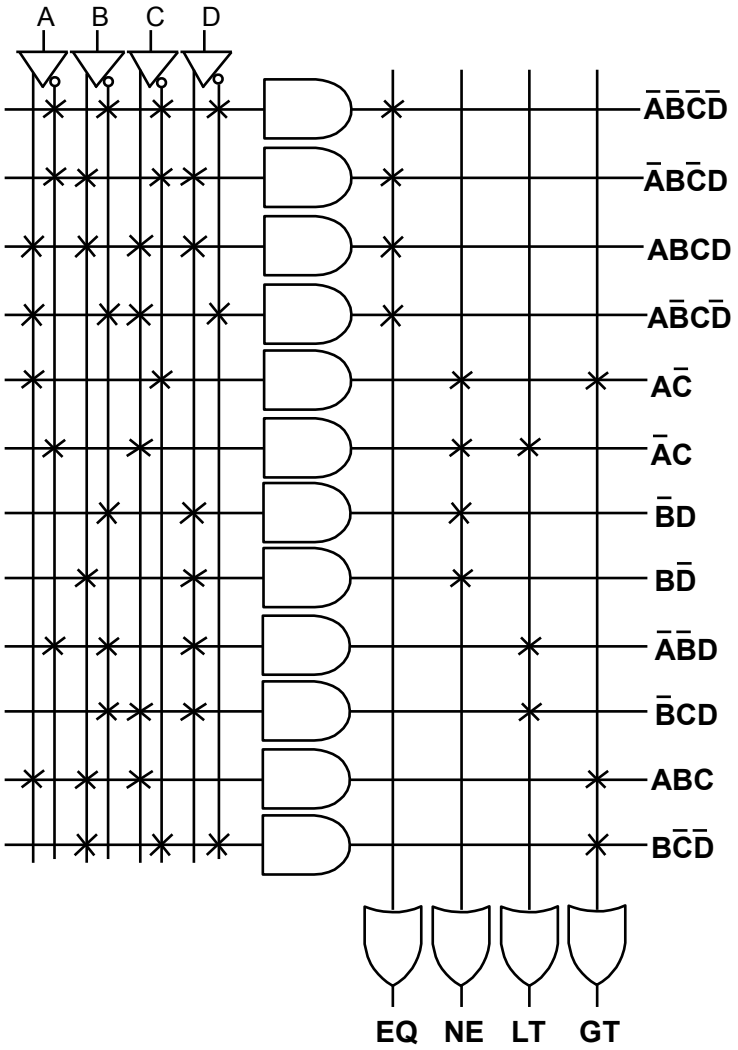
K-map for NE



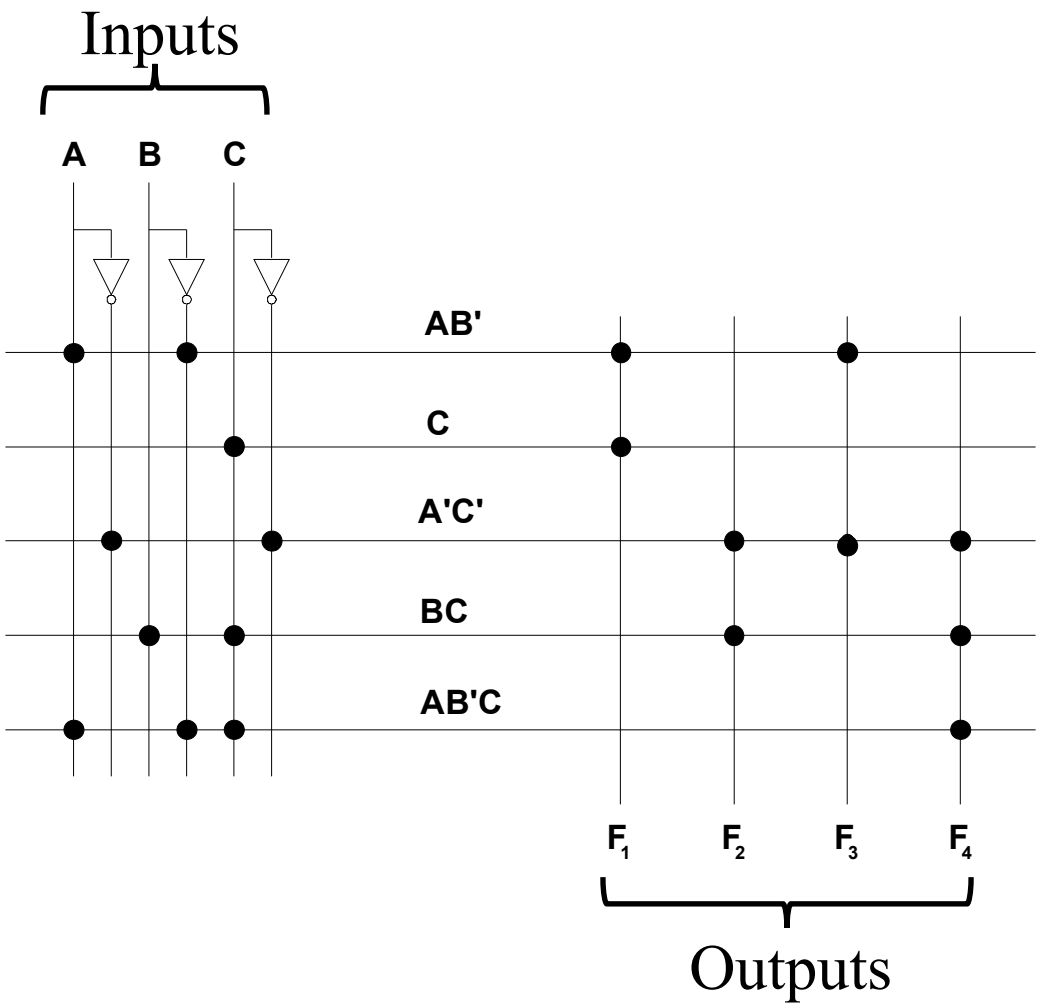
K-map for LT



K-map for GT



Another PLA Representation



PLA Example

Implement these equations:

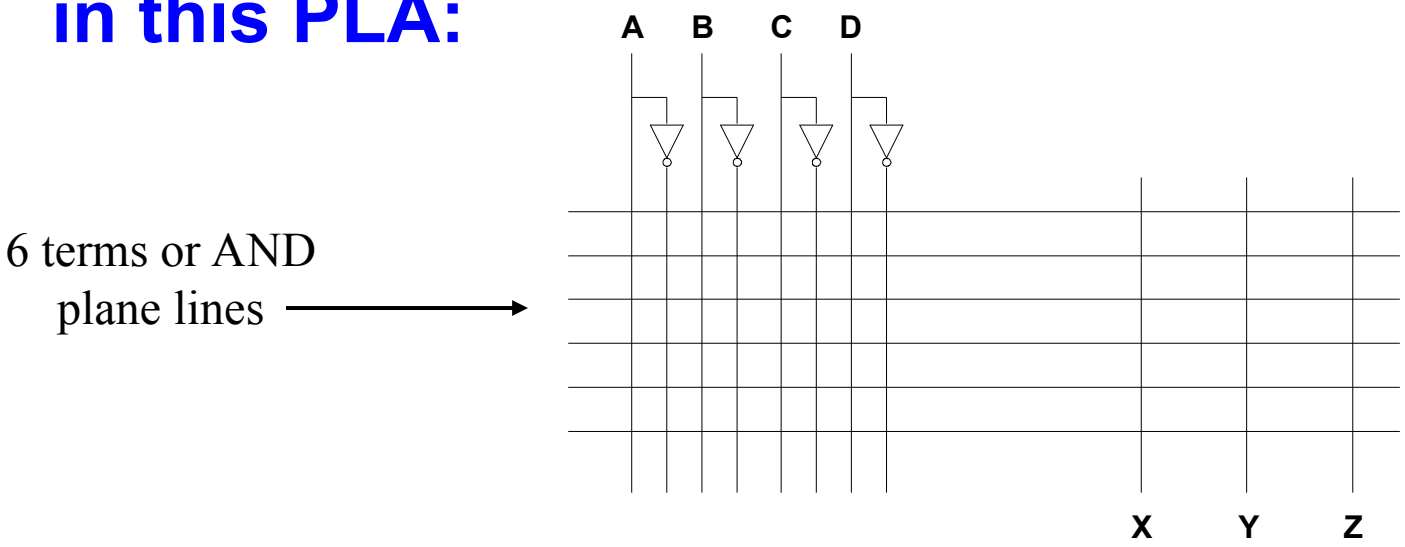
$$X = \overset{1}{ABC} + \overset{2}{B'D'} + \overset{3}{AB'D} + \overset{4}{C'D'}$$

$$Y = \overset{5}{BC} + \overset{6}{D'}$$

8 terms

$$Z = \overset{7}{CD} + \overset{2}{B'D'} + \overset{8}{A'BC}$$

in this PLA:



can we implement 8 product terms with 6 AND plane lines?

PLA Example

Implement these equations:

X = ABC + B'D' + AB'D + C'D'

Y = BC + D'

Z = CD + B'D' + A'BC

X

C D \ A B	00	01	11	10
00				
01				
11				
10				

Y

C D \ A B	00	01	11	10
00				
01				
11				
10				

Z

C D \ A B	00	01	11	10
00				
01				
11				
10				

PLA Example

Implement these equations:

X = ABC + B'D' + AB'D + C'D'

Y = BC + D'

Z = CD + B'D' + A'BC

X

C D \ A B	00	01	11	10
00	1	1	1	1
01				1
11			1	1
10	1		1	1

Y

C D \ A B	00	01	11	10
00				
01				
11				
10				

Z

C D \ A B	00	01	11	10
00				
01				
11				
10				

PLA Example

Implement these equations:

X = ABC + B'D' + AB'D + C'D'

Y = BC + D'

Z = CD + B'D' + A'BC

X

C D \ A B	00	01	11	10
00	1	1	1	1
01				1
11			1	1
10	1		1	1

Y

C D \ A B	00	01	11	10
00	1	1	1	1
01				
11		1	1	
10	1	1	1	1

Z

C D \ A B	00	01	11	10
00				
01				
11				
10				

PLA Example

Implement these equations:

X = ABC + B'D' + AB'D + C'D'

Y = BC + D'

Z = CD + B'D' + A'BC

X

C D \ A B	00	01	11	10
00	1	1	1	1
01				1
11			1	1
10	1		1	1

Y

C D \ A B	00	01	11	10
00	1	1	1	1
01				
11		1	1	
10	1	1	1	1

Z

C D \ A B	00	01	11	10
00	1			1
01				
11	1	1	1	1
10	1	1		1

PLA Example

C'D' is in **X** and **Y** and looks useful

X

C D \ A B	00	01	11	10
00	1	1	1	1
01				1
11			1	1
10	1		1	1

Y

C D \ A B	00	01	11	10
00	1	1	1	1
01				
11		1	1	
10	1	1	1	1

Z

C D \ A B	00	01	11	10
00	1			1
01				
11	1	1	1	1
10	1	1		1

PLA Example

C'D' is in **X** and **Y** and looks useful

B'D' is in all three functions

X

C D \ A B	00	01	11	10
00	1	1	1	1
01				1
11			1	1
10	1		1	1

Y

C D \ A B	00	01	11	10
00	1	1	1	1
01				
11		1	1	
10	1	1	1	1

Z

C D \ A B	00	01	11	10
00	1			1
01				
11	1	1	1	1
10	1	1		1

PLA Example

C'D' is in **X** and **Y** and looks useful

B'D' is in all three functions

A'BC and ABC cover a lot of minterms

X

C D \ A B	00	01	11	10
00	1	1	1	1
01				1
11			1	1
10	1		1	1

Y

C D \ A B	00	01	11	10
00	1	1	1	1
01				
11		1	1	
10	1	1	1	1

Z

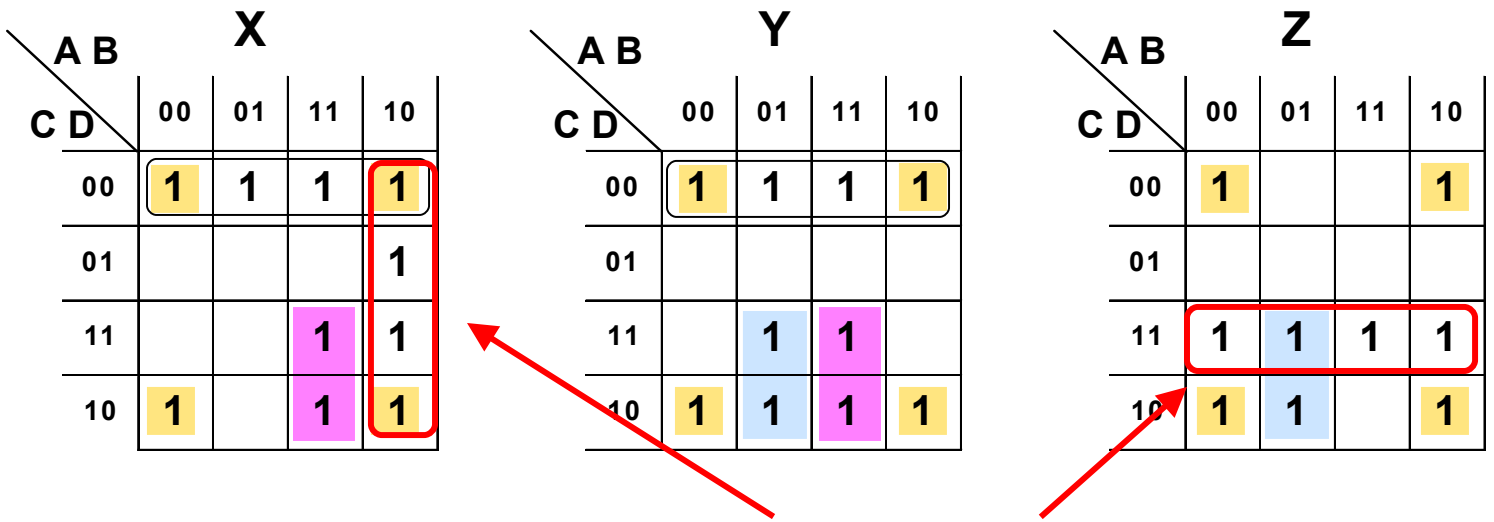
C D \ A B	00	01	11	10
00	1			1
01				
11	1	1	1	1
10	1	1		1

PLA Example

C'D' is in **X** and **Y** and looks useful

B'D' is in all three functions

A'BC and ABC cover a lot of minterms



The only ones left are AB' and CD

PLA Example

$$X = C'D' + B'D' + AB' + ABC$$

$$Y = C'D' + B'D' + ABC + A'BC$$

$$Z = B'D' + CD + A'BC$$

All of the functions are covered using only 6 product terms

X

C D \ A B	00	01	11	10
00	1	1	1	1
01				1
11			1	1
10	1		1	1

Y

C D \ A B	00	01	11	10
00	1	1	1	1
01				
11		1	1	
10	1	1	1	1

Z

C D \ A B	00	01	11	10
00	1			1
01				
11	1	1	1	1
10	1	1		1

How is this possible?

PLA Example

X = C'D' + B'D' + AB' + ABC

Y = C'D' + B'D' + ABC + A'BC

Z = B'D' + CD + A'BC

Product Term	Input A	B	C	D	Output X	Y	Z
C'D'							
B'D'							
AB'							
ABC							
A'BC							
CD							

PLA Example

X = C'D' + B'D' + AB' + ABC

Y = C'D' + B'D' + ABC + A'BC

Z = B'D' + CD + A'BC

Product Term	Input A	B	C	D	Output X	Y	Z
C'D'	-	-	0	0			
B'D'	-	0	-	0			
AB'	1	0	-	-			
ABC	1	1	1	-			
A'BC	0	1	1	-			
CD	-	-	1	1			

PLA Example

X = C'D' + B'D' + AB' + ABC

Y = C'D' + B'D' + ABC + A'BC

Z = B'D' + CD + A'BC

Product Term	Input A	B	C	D	Output X	Y	Z
C'D'	-	-	0	0	1	1	0
B'D'	-	0	-	0	1	1	1
AB'	1	0	-	-	1	0	0
ABC	1	1	1	-	1	1	0
A'BC	0	1	1	-	0	1	1
CD	-	-	1	1	0	0	1

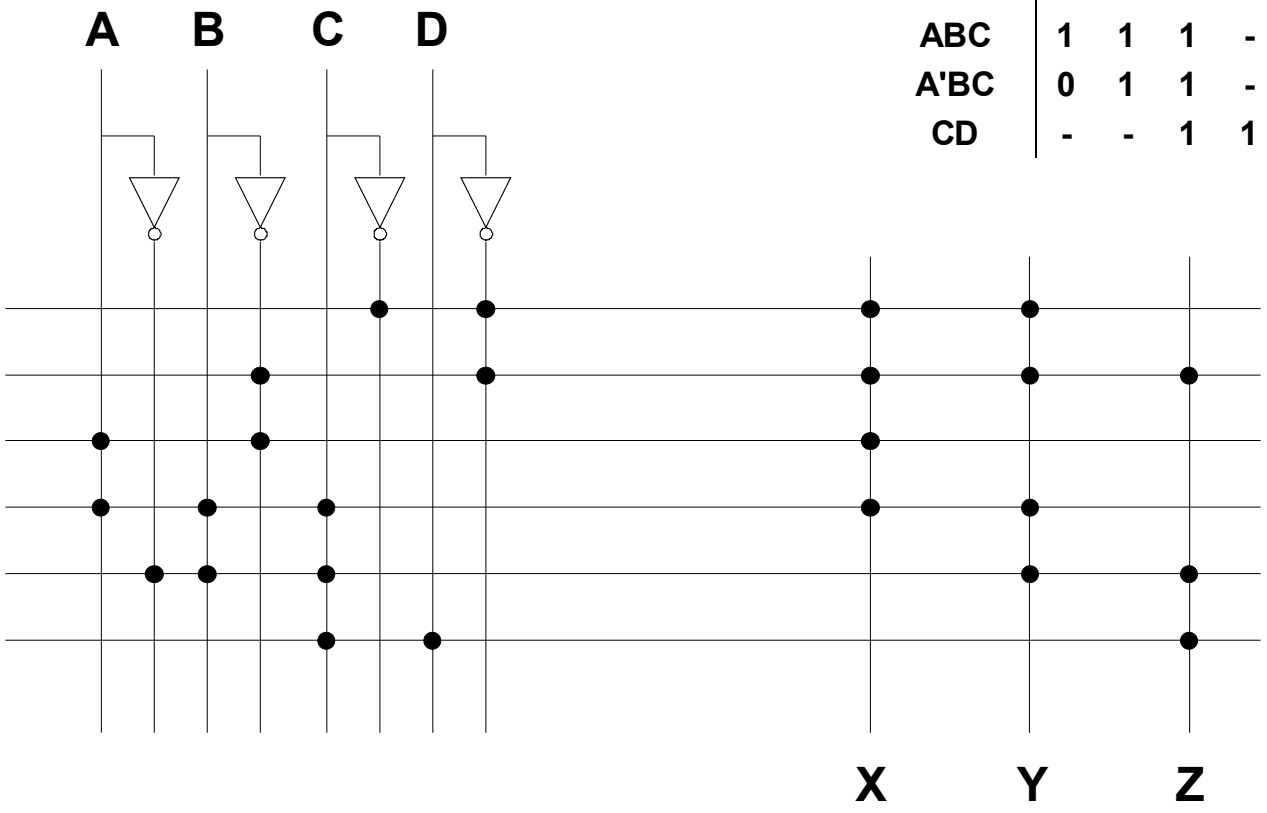
PLA Example

X = C'D' + B'D' + AB' + ABC

Y = C'D' + B'D' + ABC + A'BC

Z = B'D' + CD + A'BC

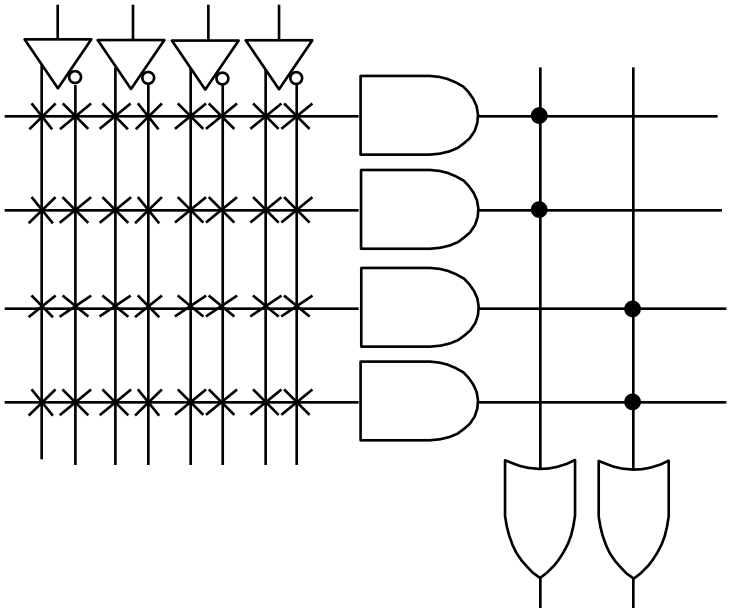
Product Term	Input A	B	C	D	Output X	Y	Z
C'D'	-	-	0	0	1	1	0
B'D'	-	0	-	0	1	1	1
AB'	1	0	-	-	1	0	0
ABC	1	1	1	-	1	1	0
A'BC	0	1	1	-	0	1	1
CD	-	-	1	1	0	0	1



PALs and PLAs

What is difference between Programmable Array Logic (PAL) and Programmable Logic Array (PLA)?

PAL concept — implemented by Monolithic Memories
AND array is programmable, OR array is fixed at fabrication



A given column of the OR array has access to only a subset of the possible product terms

PLA concept — Both AND and OR arrays are programmable

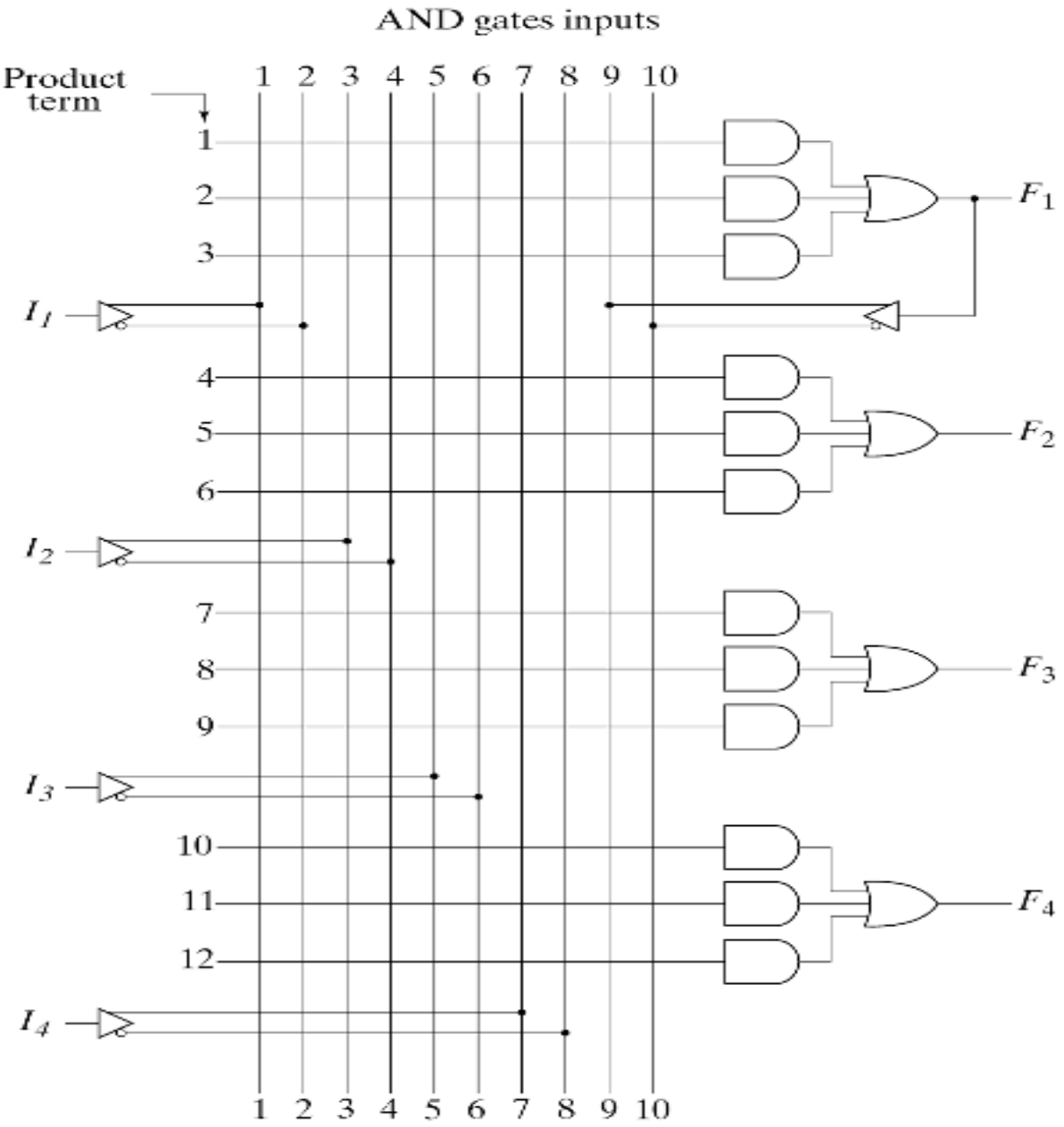


Fig. 7-16 PAL with Four Inputs, Four Outputs, and Three-Wide AND-OR Structure

$$X = A\bar{C} + B\bar{C} + \bar{B}C + \bar{B}D$$

$$Y = AC + AD + \bar{A}\bar{D} + \bar{B}\bar{C}$$

$$Z = AB + \bar{A}\bar{B}$$

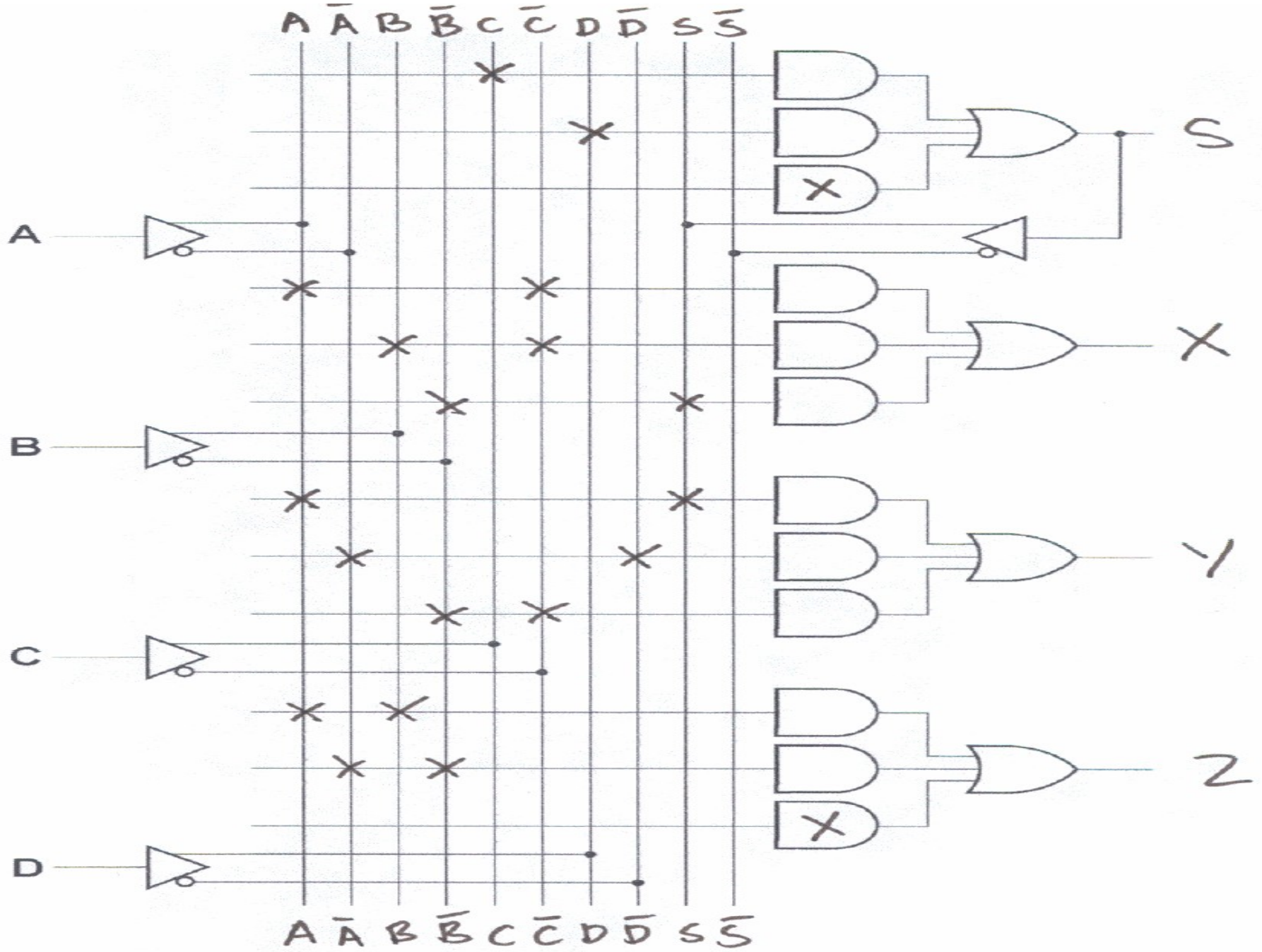
$$X = A\bar{C} + B\bar{C} + \bar{B}(C + D)$$

$$Y = A(C + D) + \bar{A}\bar{D} + \bar{B}\bar{C}$$

$$Z = AB + \bar{A}\bar{B}$$

$$S = C + D$$

	AND Inputs					Output Function
	A	B	C	D	S	
1	–	–	1	–	–	$S = C + D$
2	–	–	–	1	–	
3	10	10	10	10	10	
4	1	–	0	–	–	$X = A\bar{C} + B\bar{C} + \bar{B}S$
5	–	1	0	–	–	
6	–	0	–	–	1	
7	1	–	–	–	1	$Y = AS + \bar{A}\bar{D} + \bar{B}\bar{C}$
8	0	–	–	0	–	
9	–	0	0	-	–	
10	1	1	–	–	–	$Z = AB + \bar{A}\bar{B}$
11	0	0	–	–	–	
12	10	10	10	10	10	


A Ā B B̄ C C̄ D D̄ S S̄
A
B
C
D
S
X
Y
Z
A Ā B B̄ C C̄ D D̄ S S̄

PALs and PLAs

- Of the two organizations the PLA is the most flexible
  - One PLA can implement a huge range of logic functions
  - BUT many pins; large package, higher cost
- PALs are more restricted / you trade number of OR terms vs number of outputs
  - Many device variations needed
  - Each device is cheaper than a PLA

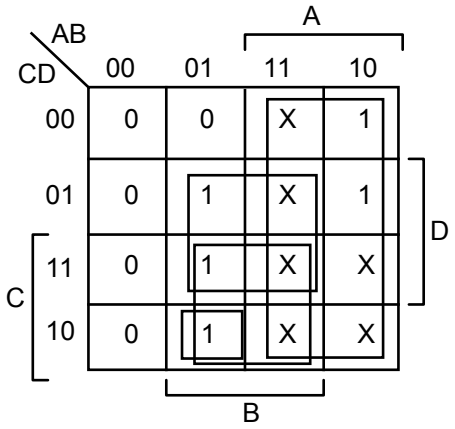
PAL Logic Implementation

Design Example: BCD to Gray Code Converter

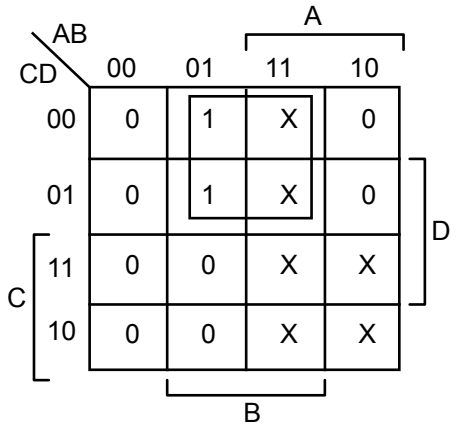
Truth Table

A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

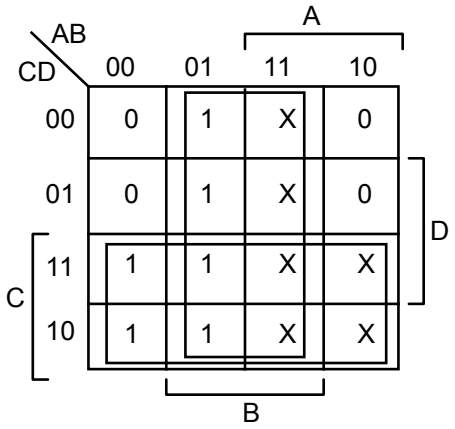
K-maps



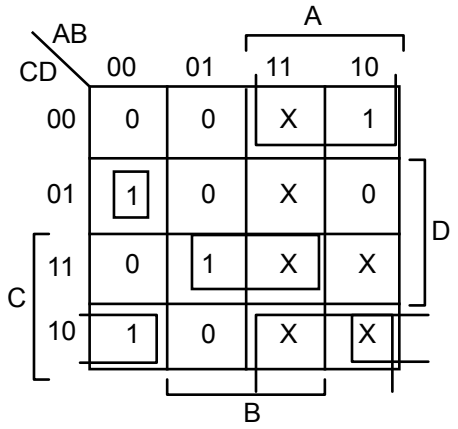
K-map for W



K-map for X



K-map for Y



K-map for Z

Minimized Functions:

$W = A + B\,D + B\,C$

$X = B\,\overline{C}$

$Y = B + C$

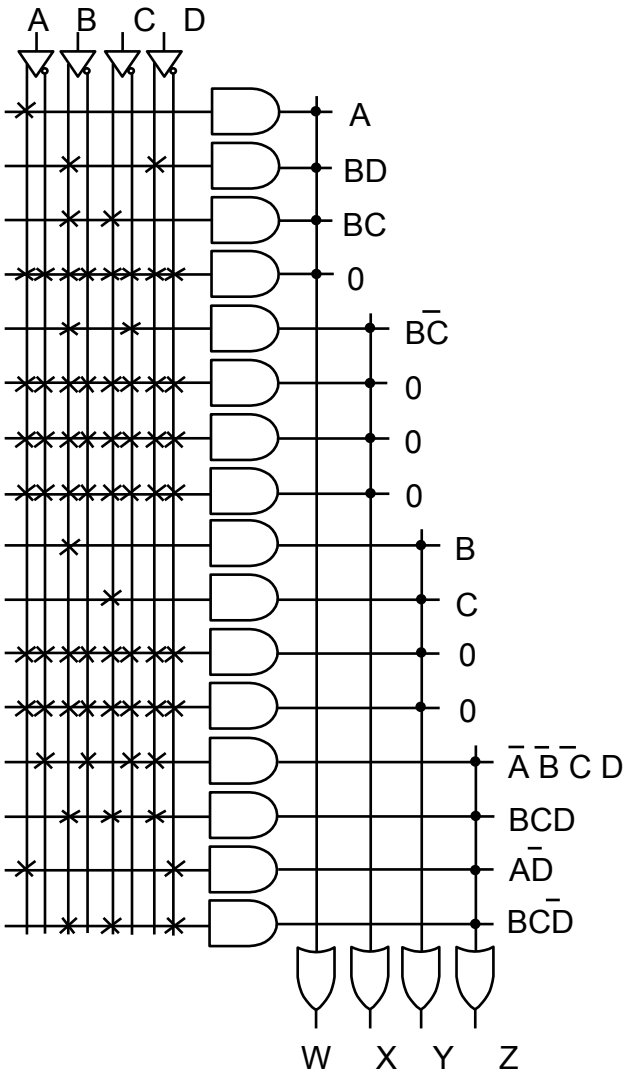
$Z = \overline{A}\,\overline{B}\,\overline{C}\,D + B\,C\,D + A\,\overline{D} + \overline{B}\,C\,\overline{D}$

PAL Logic Implementation

Programmed PAL:

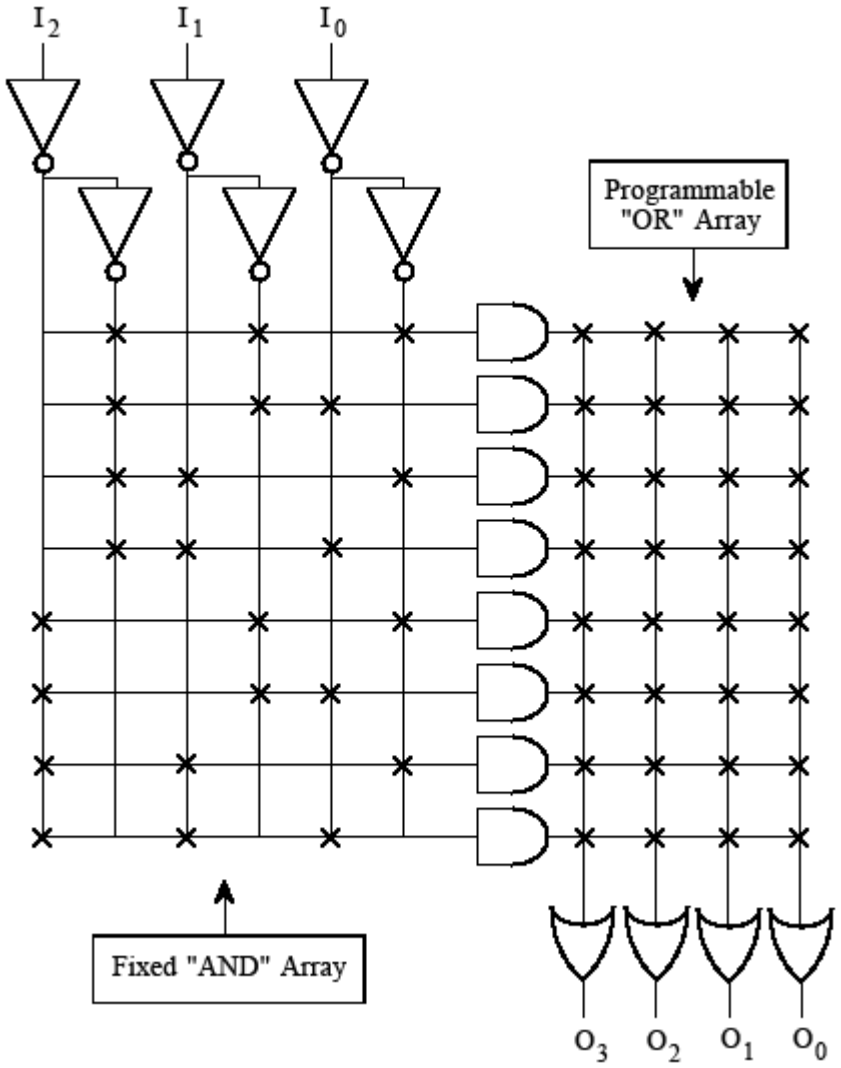
Minimized Functions:

$W = A + B\,D + B\,C$
$X = B\,\overline{C}$
$Y = \overline{B} + \overline{C}$
$Z = \overline{A}\,\overline{B}\,\overline{C}\,D + B\,C\,D + A\,\overline{D} + \overline{B}\,C\,\overline{D}$

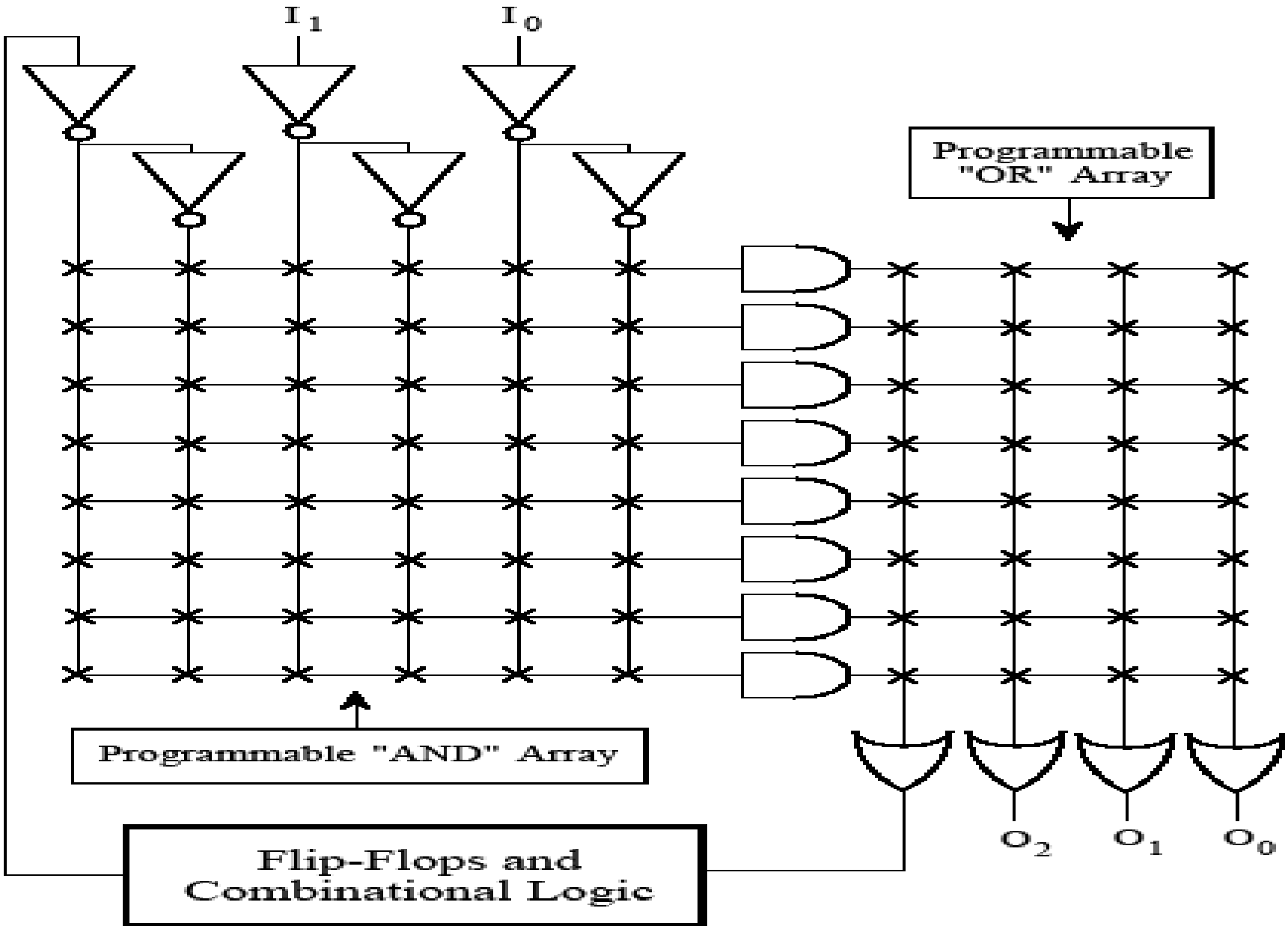


4 product terms per each OR gate

PROM with 8 Words X 4 Bits



PLS with 2 Inputs and 3 Outputs



FPGA AND CPLD

1. FPGA - Field-Programmable Gate Array.
2. CPLD - Complex Programmable Logic Device
3. FPGA and CPLD is an advance PLD.
4. Support thousands of gate where as PLD only support hundreds of gates.

What is an FPGA?

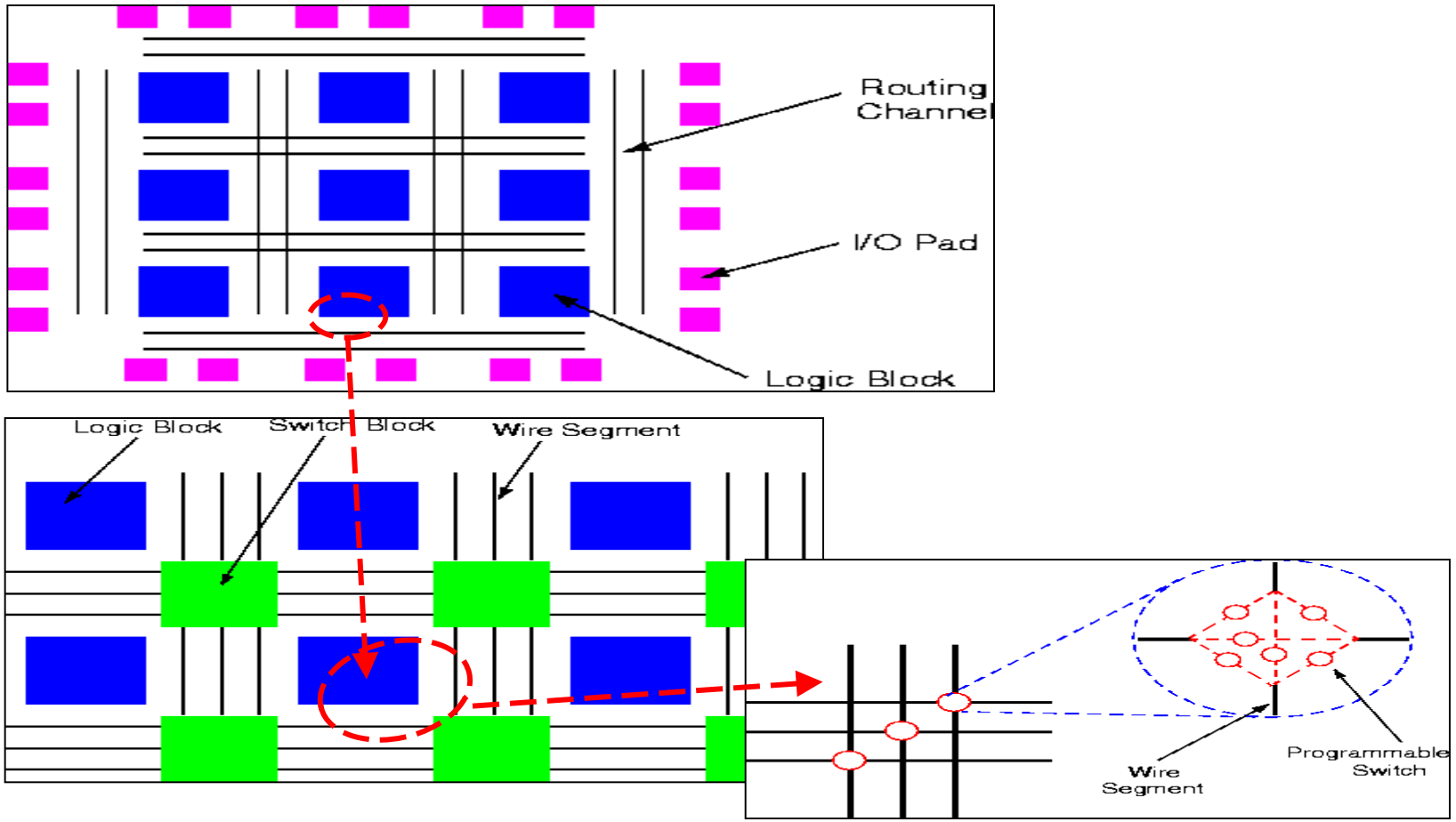
- Before the advent of programmable logic, custom logic circuits were built at the board level using standard components, or at the gate level in expensive application-specific (custom) integrated circuits.

- FPGA is an integrated circuit that contains many (64 to over 10,000) identical logic cells that can be viewed as standard components. Each logic cell can independently take on any one of a limited set of personalities.

- Individual cells are interconnected by a matrix of wires and programmable switches. A user's design is implemented by specifying the simple logic function for each cell and selectively closing the switches in the interconnect matrix.

- Array of logic cells and interconnect form a fabric of basic building blocks for logic circuits. Complex designs are created by combining these basic blocks to create the desired circuit

FPGA architecture



What does a logic cell do?

- The logic cell architecture varies between different device families.
- Each logic cell combines a few binary inputs (typically between 3 and 10) to one or two outputs according to a Boolean logic function specified in the user program .
- In most families, the user also has the option of registering the combinatorial output of the cell, so that clocked logic can be easily implemented.
- Cell's combinatorial logic may be physically implemented as a small look-up table memory (LUT) or as a set of multiplexers and gates.
- LUT devices tend to be a bit more flexible and provide more inputs per cell than multiplexer cells at the expense of propagation delay.

what does 'Field Programmable' mean?

- Field Programmable means that the FPGA's function is defined by a user's program rather than by the manufacturer of the device.
- A typical integrated circuit performs a particular function defined at the time of manufacture. In contrast, the FPGA's function is defined by a program written by someone other than the device manufacturer.
- Depending on the particular device, the program is either 'burned' in permanently or semi-permanently as part of a board assembly process, or is loaded from an external memory each time the device is powered up.
- This user programmability gives the user access to complex integrated designs without the high engineering costs associated with application specific integrated circuits.

How are FPGA programs created?

- Individually defining the many switch connections and cell logic functions would be a daunting task.
- This task is handled by special software. The software translates a user's schematic diagrams or textual hardware description language code then places and routes the translated design.
- Most of the software packages have hooks to allow the user to influence implementation, placement and routing to obtain better performance and utilization of the device.
- Libraries of more complex function macros (eg. adders) further simplify the design process by providing common circuits that are already optimized for speed or area.

FPGA

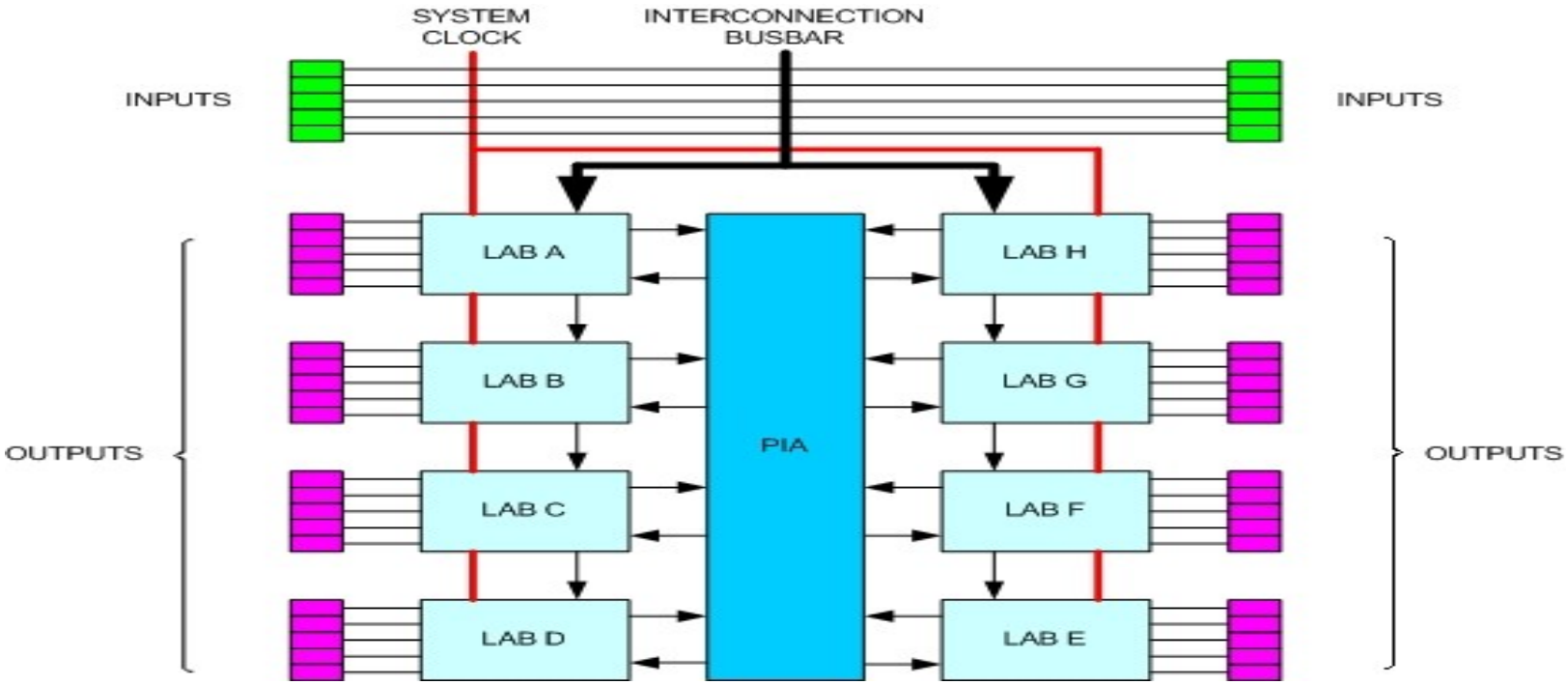
- FPGA applications:-
  i. DSP
  ii. Software-defined radio
  iii. Aerospace
  iv. Defense system
  v. ASIC Prototyping
  vi. Medical Imaging
  vii. Computer vision
  viii. Speech Recognition
  ix. Cryptography
  x. Bioinformatic
  xi. And others.

CPLD

1. Complexity of CPLD is between FPGA and PLD.
2. CPLD featured in common PLD:-
   i. Non-volatile configuration memory – does not need an external configuration PROM.
   ii. Routing constraints. Not for large and deeply layered logic.
3. CPLD featured in common FPGA:-
   i. Large number of gates available.
   ii. Can include complicated feedback path.
4. CPLD application:-
   i. Address coding
   ii. High performance control logic
   iii. Complex finite state machines

CPLD

5. CPLD architecture:-



LAB – Logic Array Block / uses PALs

PIA – Programmable Interconnect Array