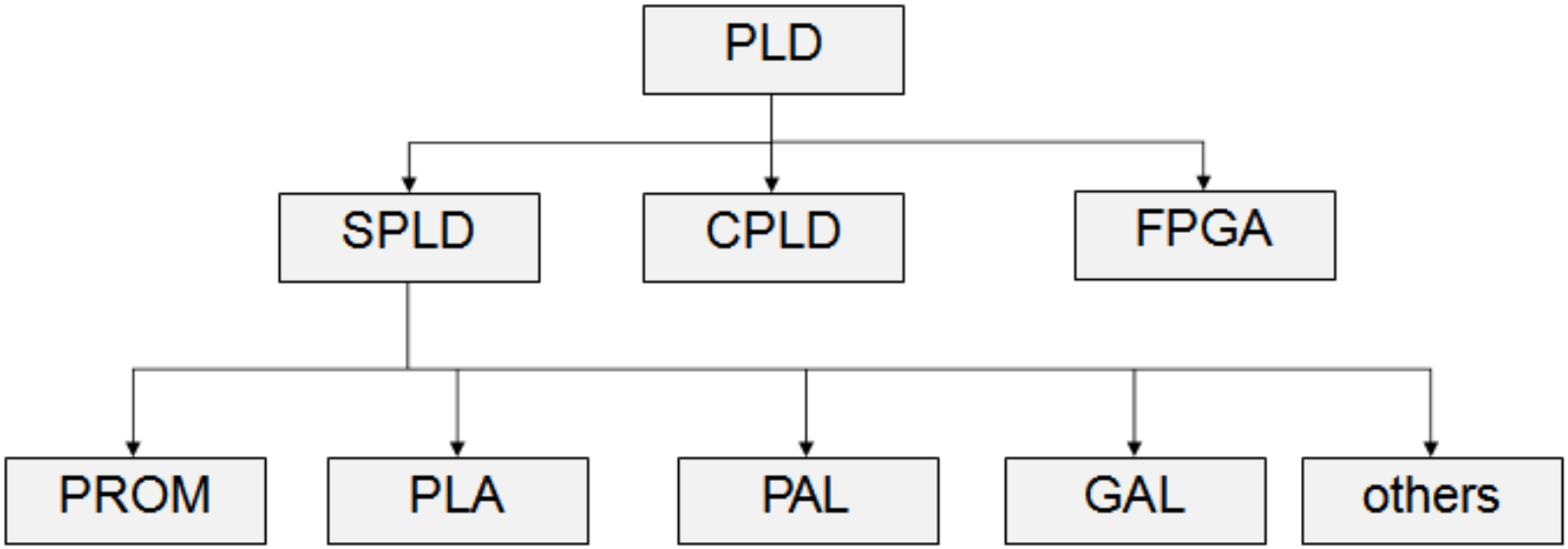
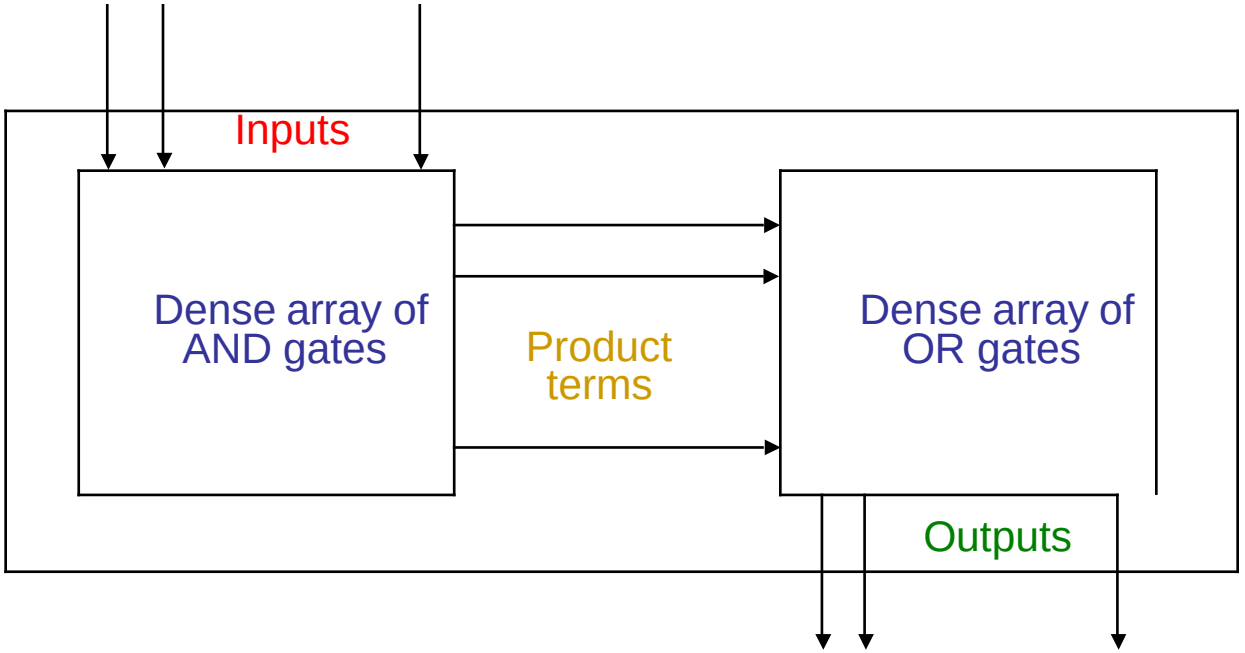
Programmable Logic Devices

PLD’s



Prgrammable Logic Organization

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

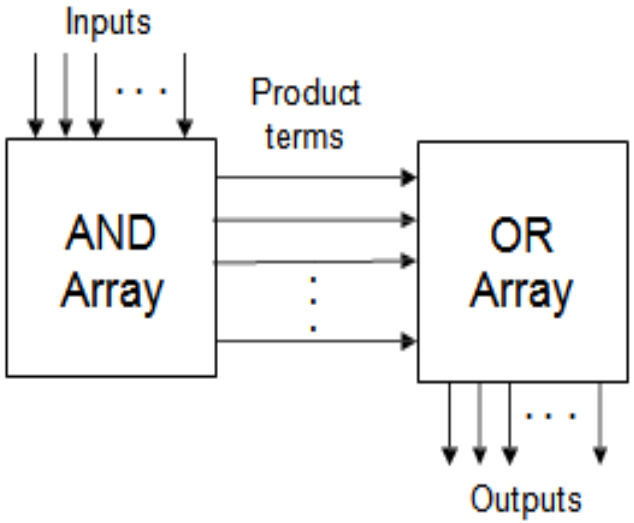


Programmable Array Block Diagram for Sum of Products Form

Programmable Logic Devices (PLD)

- PROM: Programmable Read Only Memory
- PLA: Programmable Logic Array
- PAL: Programmable Array Logic

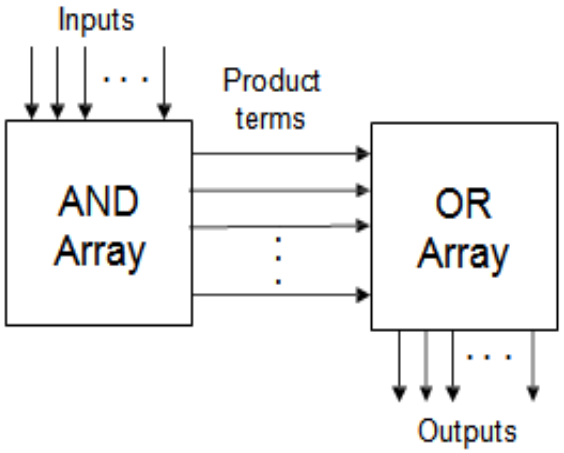
- GAL: Generic Array Logic

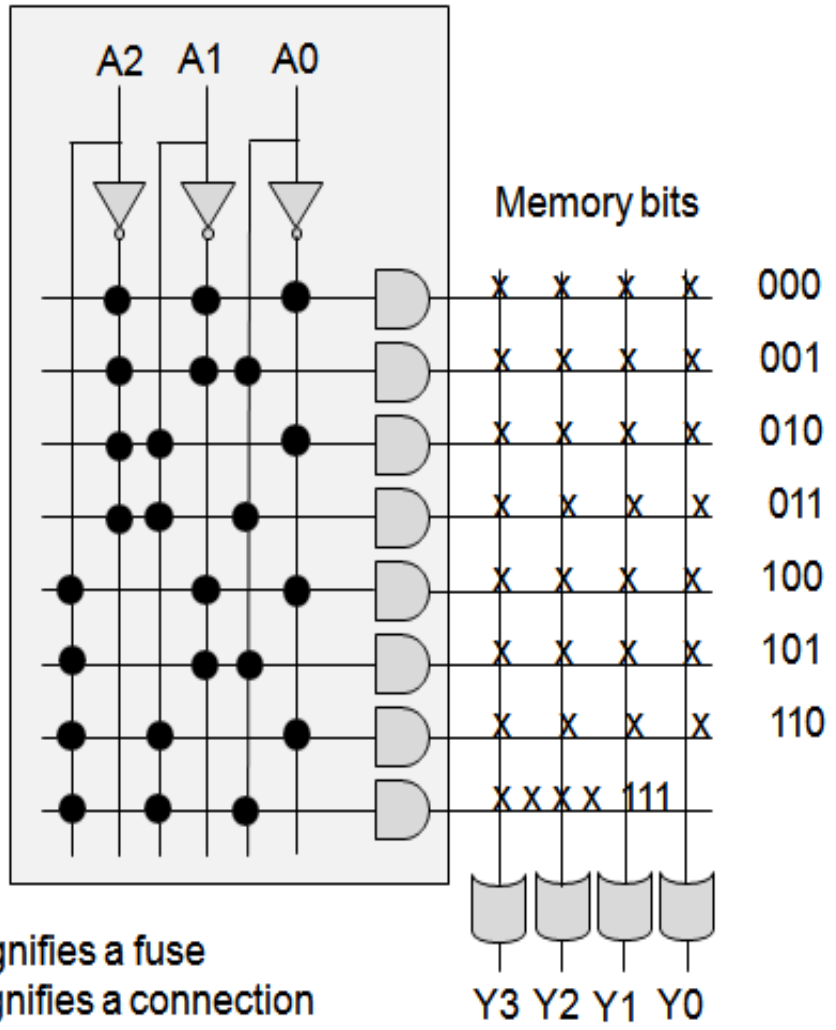


PROM

- First simple PLD

Can be viewed as a **fixed** array of **AND** functions driving a **programmable** array of **OR** functions



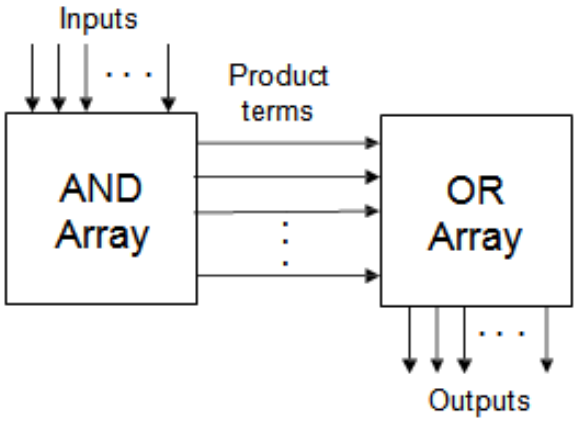


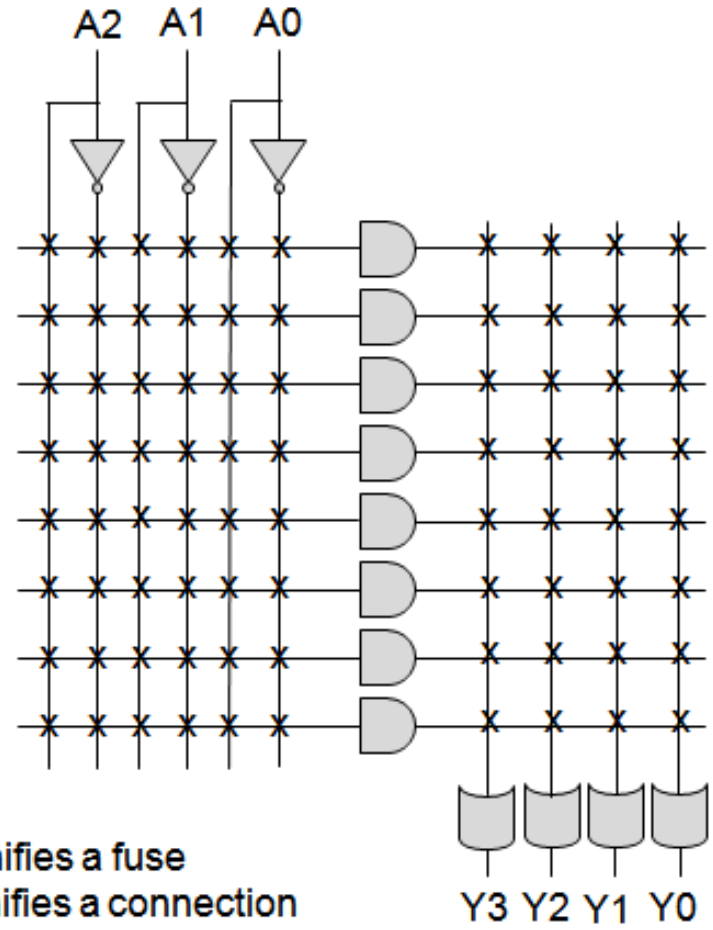
X signifies a fuse

● signifies a connection

Programmable Logic Array (PLA)

- Programmable AND & OR gate array





X signifies a fuse

● signifies a connection

Programmable Array Logic (PAL)

- Programmable AND & fixed OR

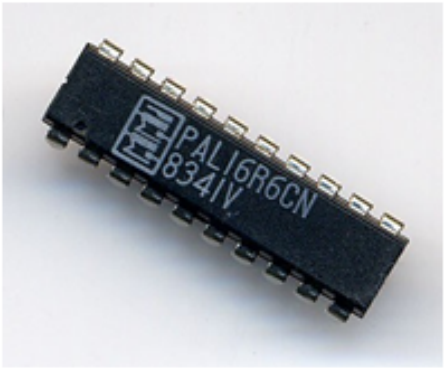
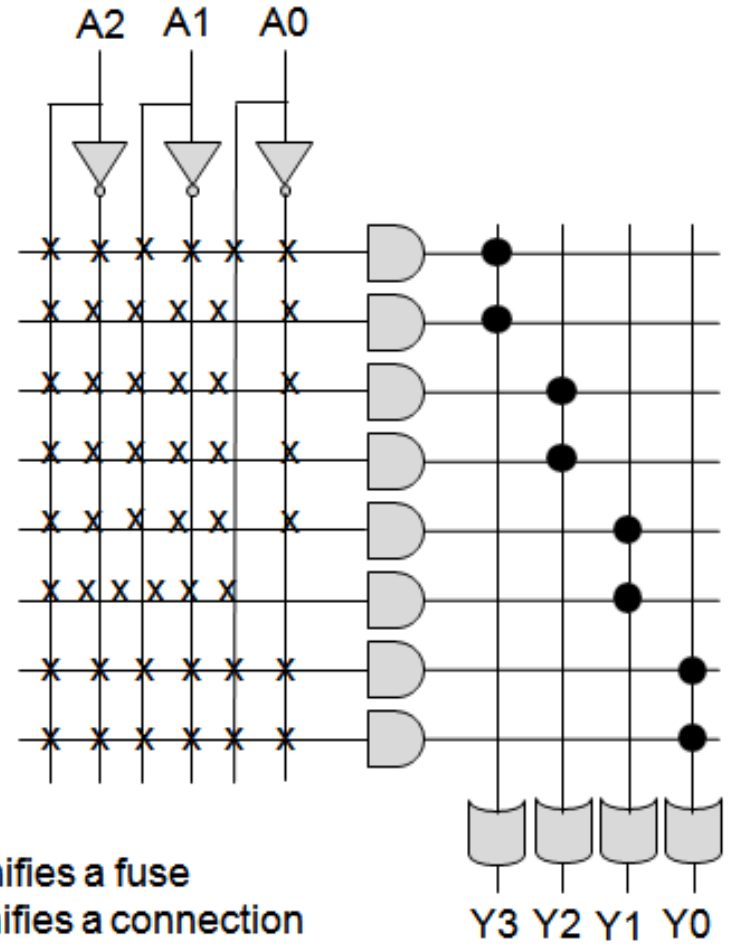
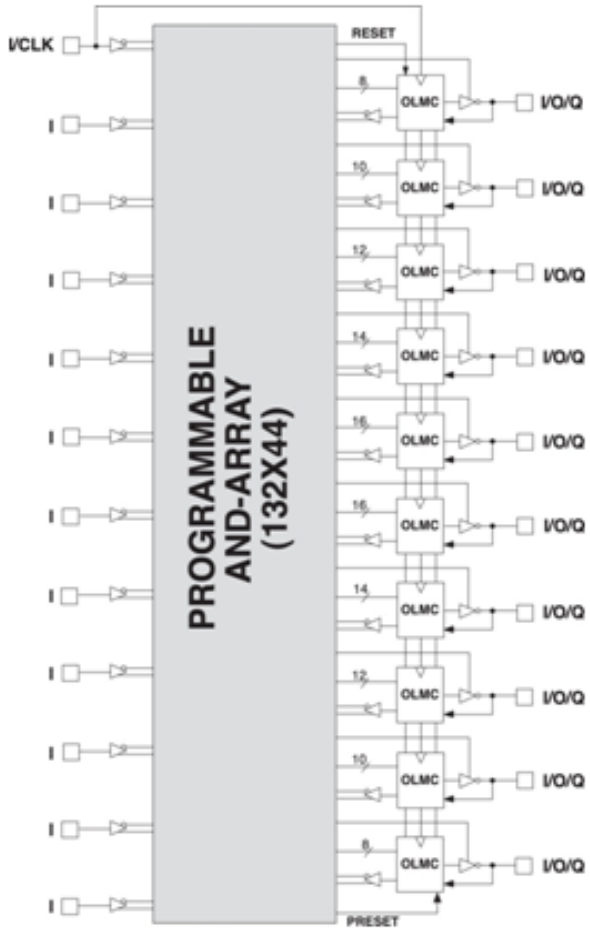


Photo: Michael Holley, 2006.
Accessed from: http://commons.wikimedia.org



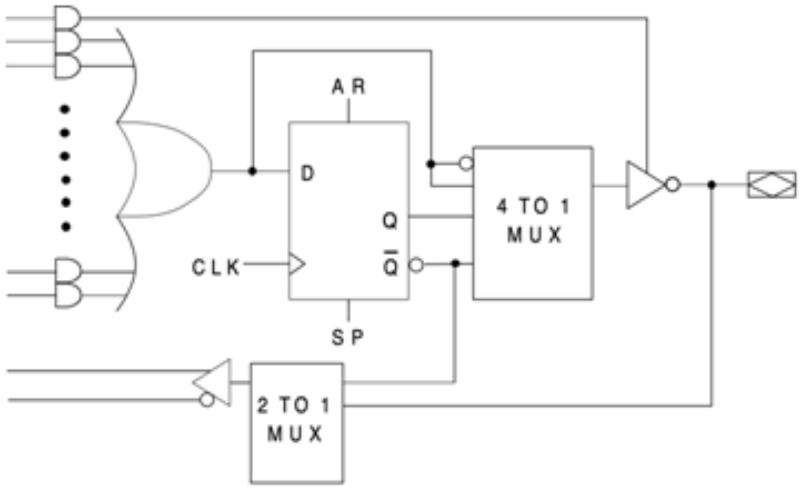
X signifies a fuse
● signifies a connection

GAL



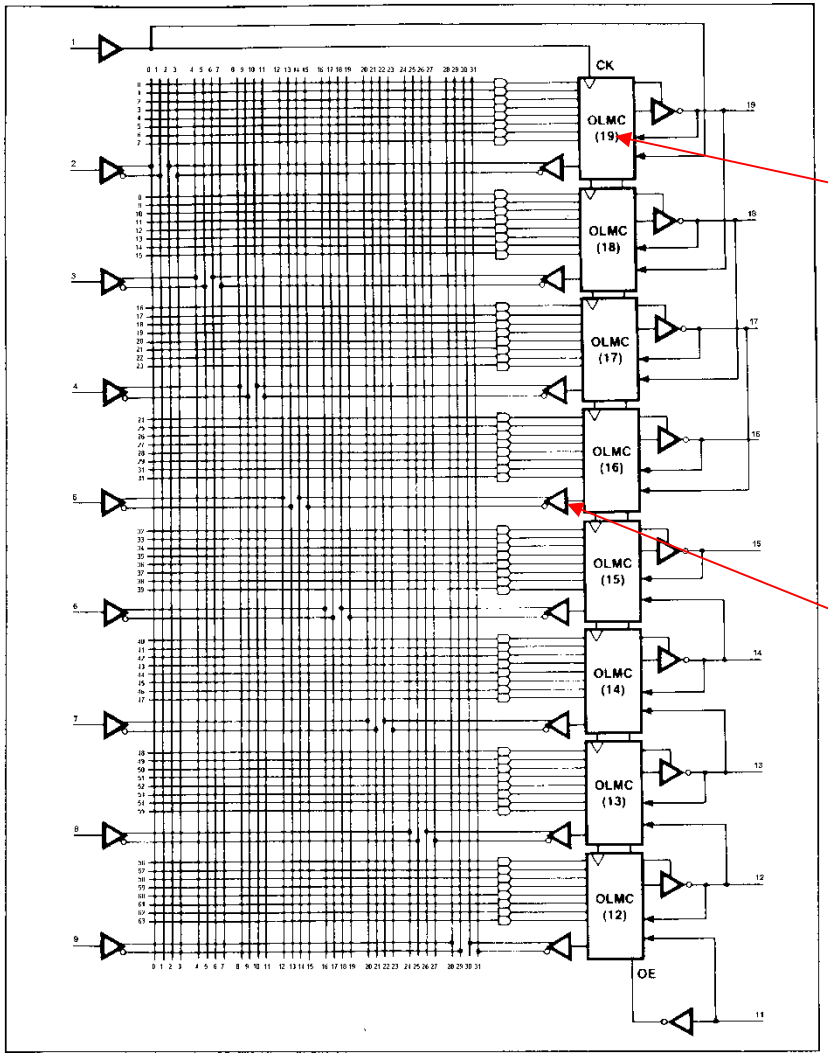
Generic Array Logic

- PAL + output macrocell
  - w/ e.g flip-flop & mux



GAL22V10 OUTPUT LOGIC MACROCELL (OLMC)

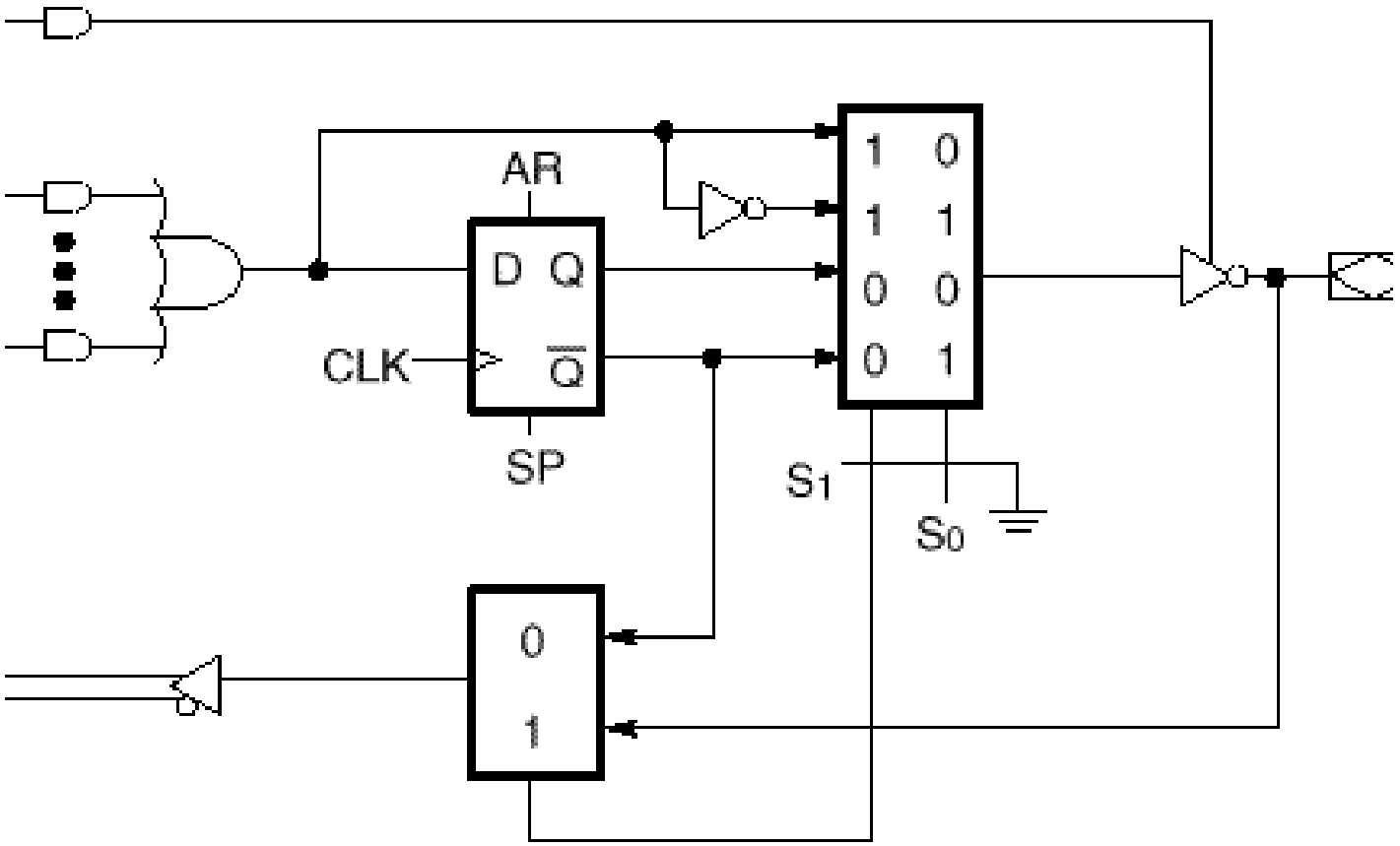
GAL CPLD



OLMC (Output Logic MacroCell) has OR, FF, output multiplexer and I/O control logic.

Note that OLMC output is fed back to input matrix for use in other OLMCs.

GAL22V10 OLMC Structure



PLA Logic Implementation

Equations

Example:

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

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

$F2 = B\,C + A\,B$

$F3 = B\,C + A$

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
$B\,C$	-	1	1	1	0	1	0
A	1	-	-	1	0	0	1

Reuse of terms

Input Side:

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

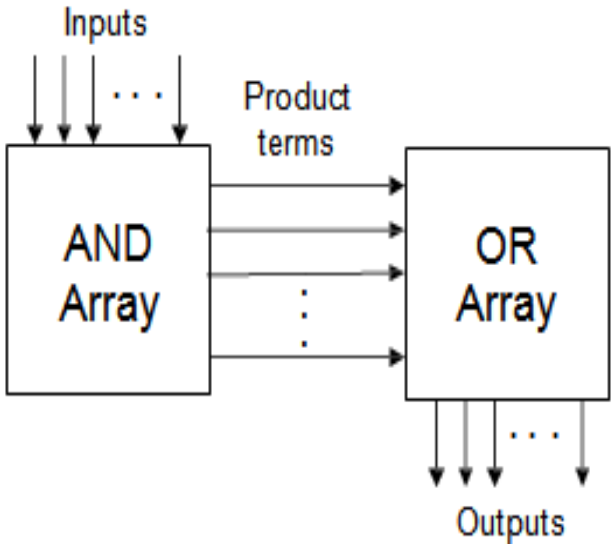
Output Side:

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

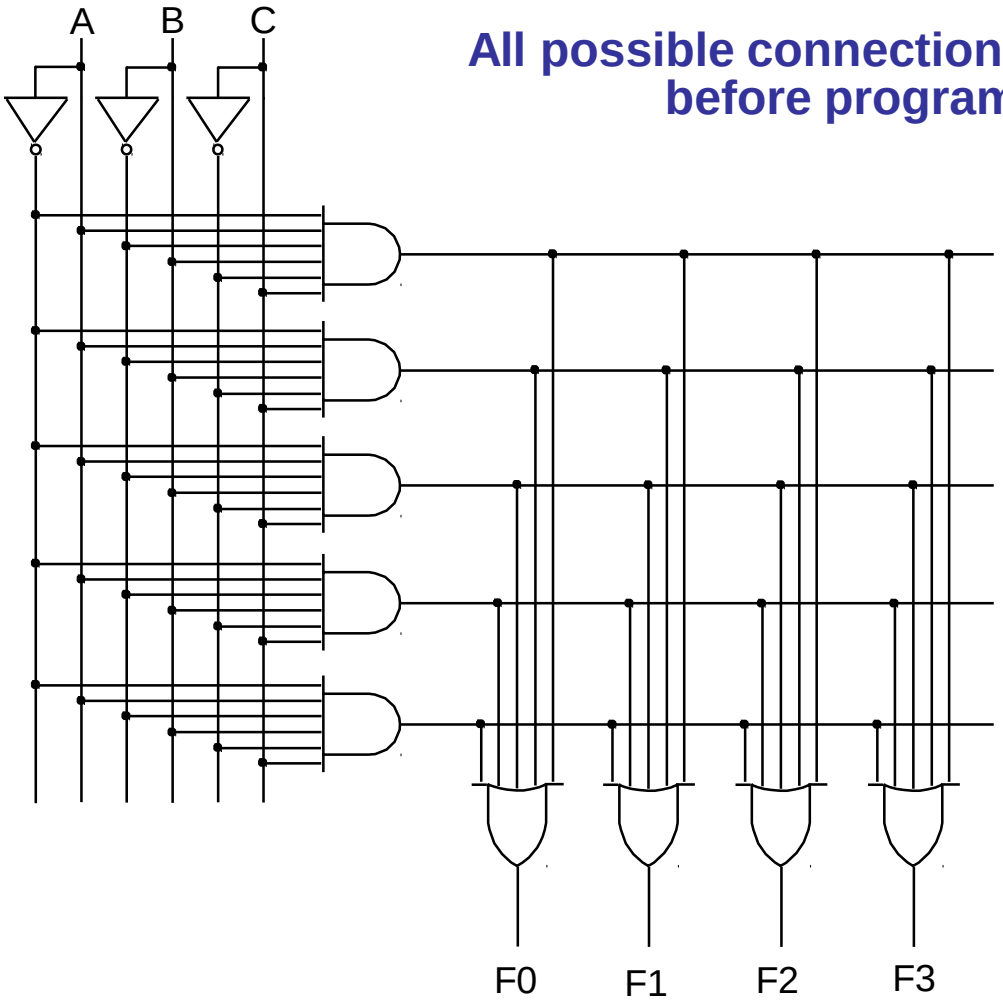
Programmable Logic Devices (PLD)

- PROM: Programmable Read Only Memory
- PLA: Programmable Logic Array
- PAL: Programmable Array Logic

- GAL: Generic Array Logic



PLA

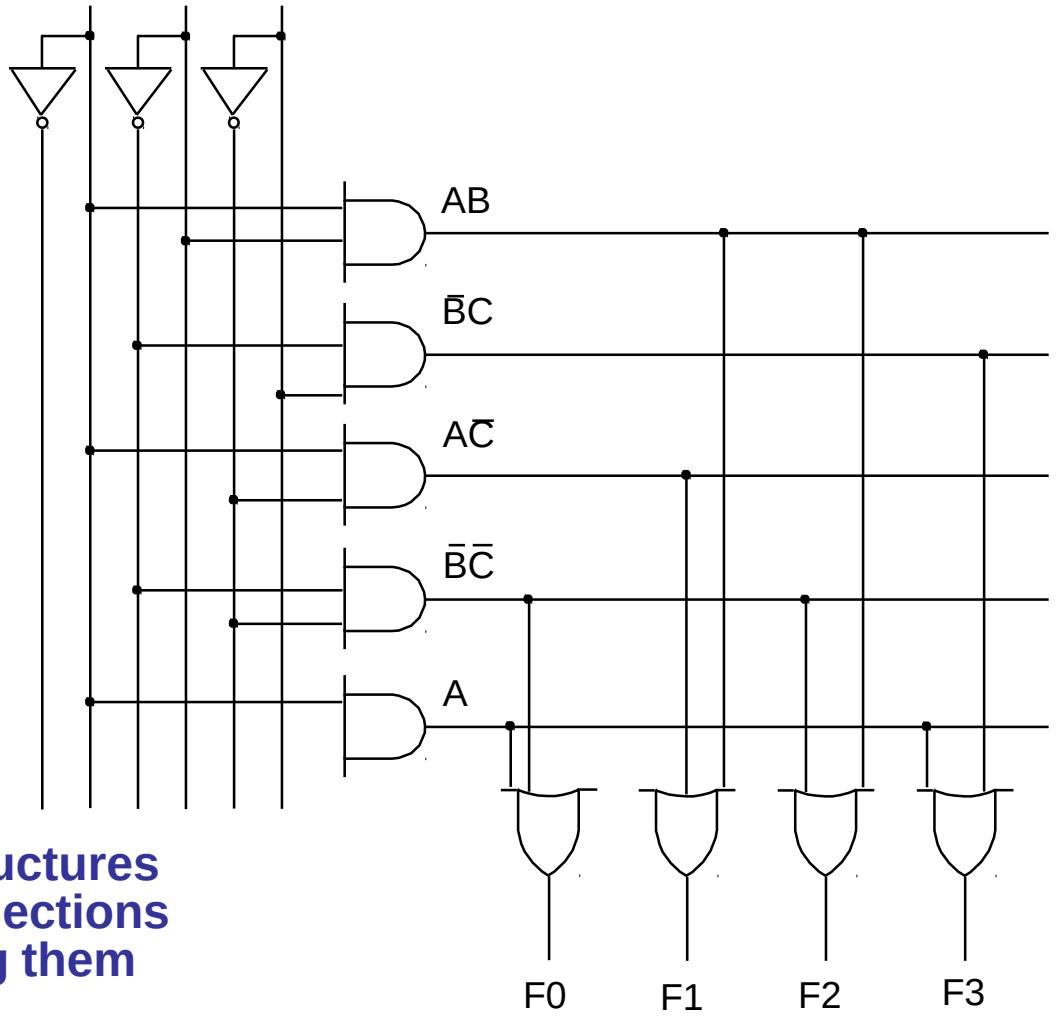


All possible connections are available before programming

Example Continued -

Unwanted connections are "blown"

Programmed part



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

alternative

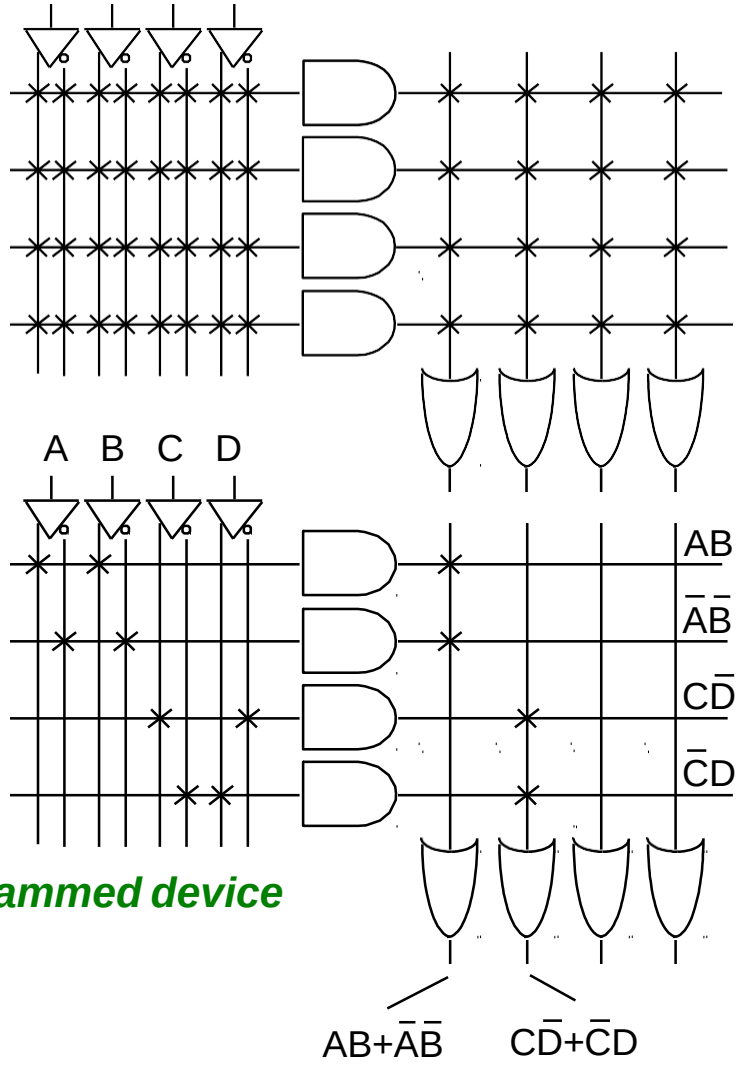
Short-hand notation so we don't have to draw all the wires!

X at junction indicates a connection

Notation for implementing

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

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



Programmed device

PLA example

Design example

Multiple functions of A, B, C

$F1 = A\,B\,C$

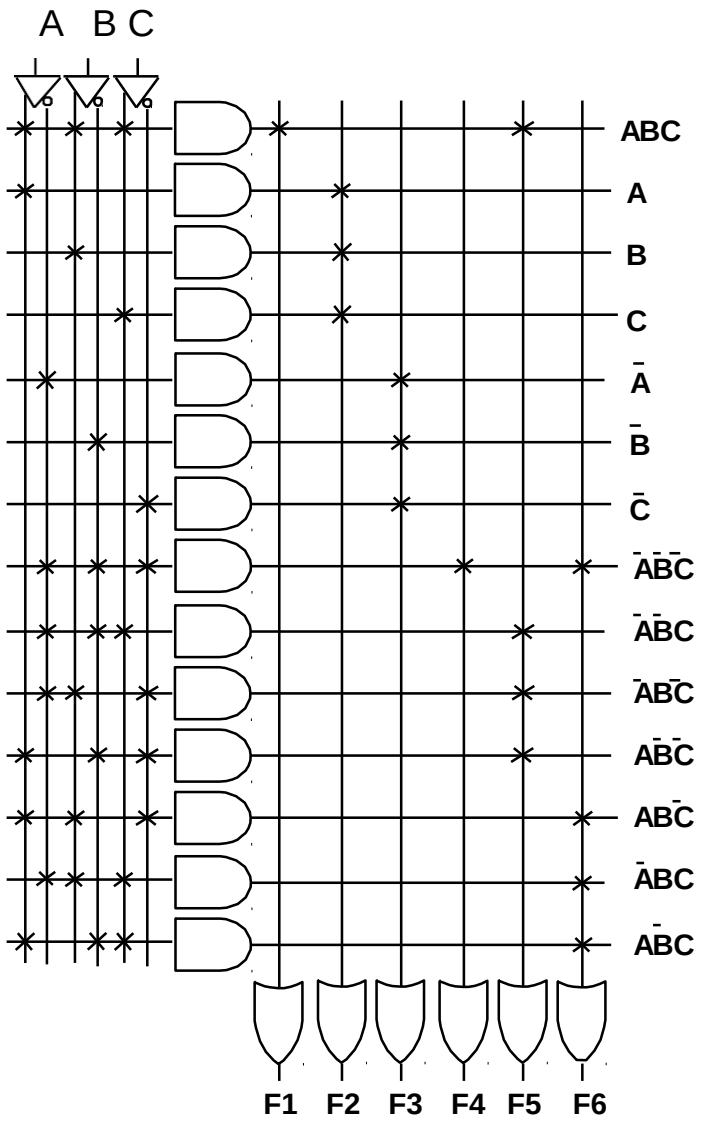
$F2 = A + B + C$

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

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

$F5 = A \oplus B \oplus C$

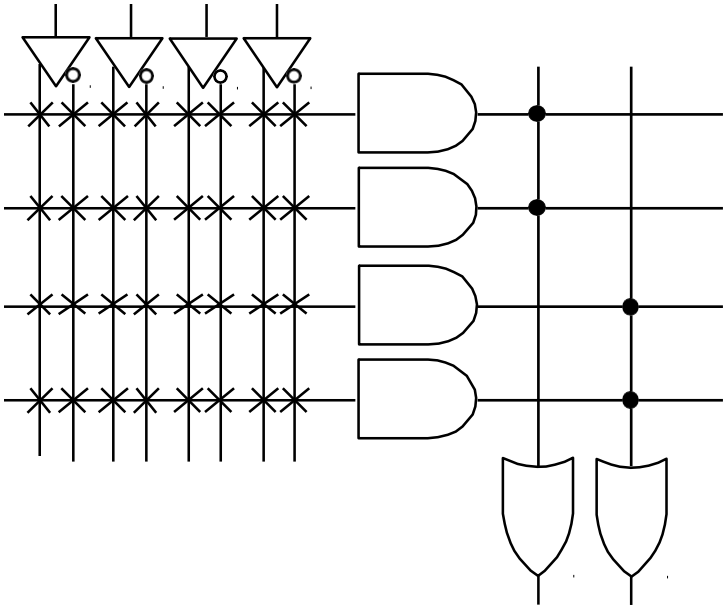
$F6 = \overline{A \oplus B \oplus C}$



PALs and PLA

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

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

Programmed PAL:

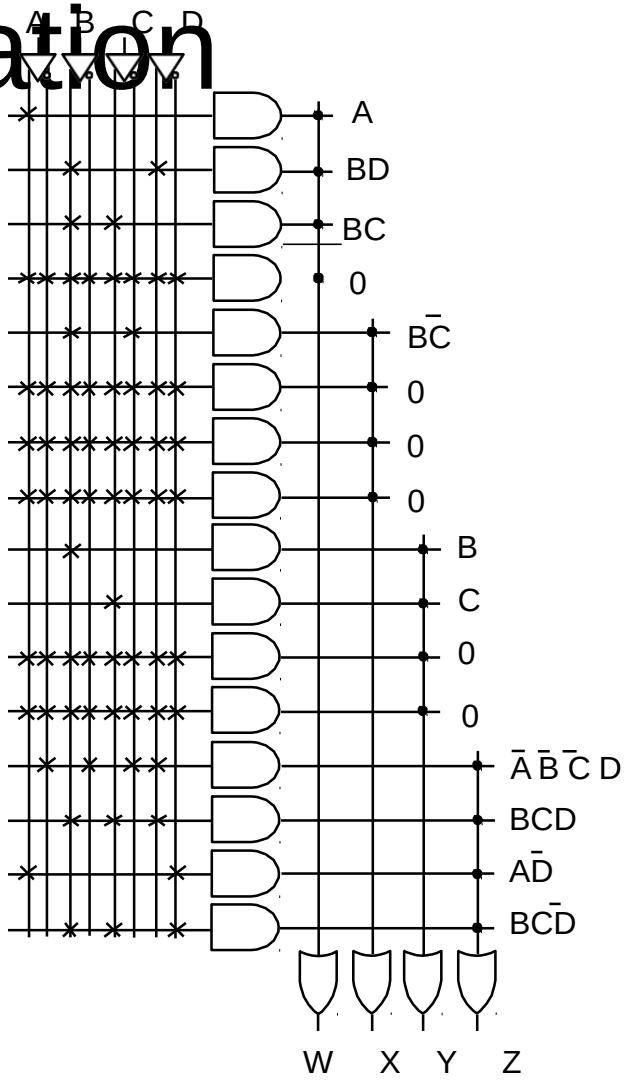
Minimized Functions:

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

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

$Y = \overline{B} + \overline{C}$

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



4 product terms per each OR gate

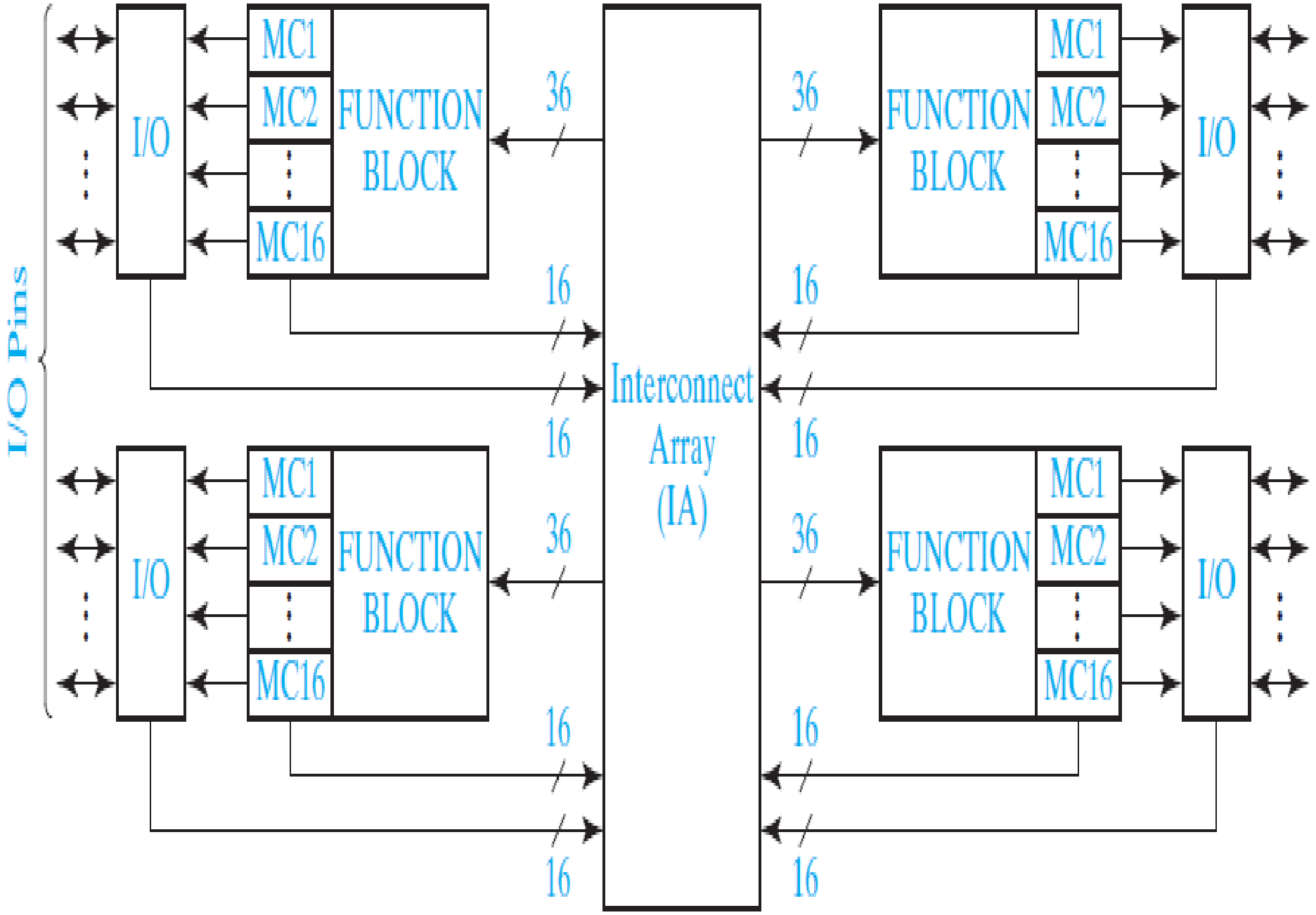
CPLD & FPGA

Contents:

- ✓ Introduction
- ✓ Architecture
- ✓ Comparison between CPLD & FPGA

COMPLEX PROGRAMMABLE LOGIC DEVICES

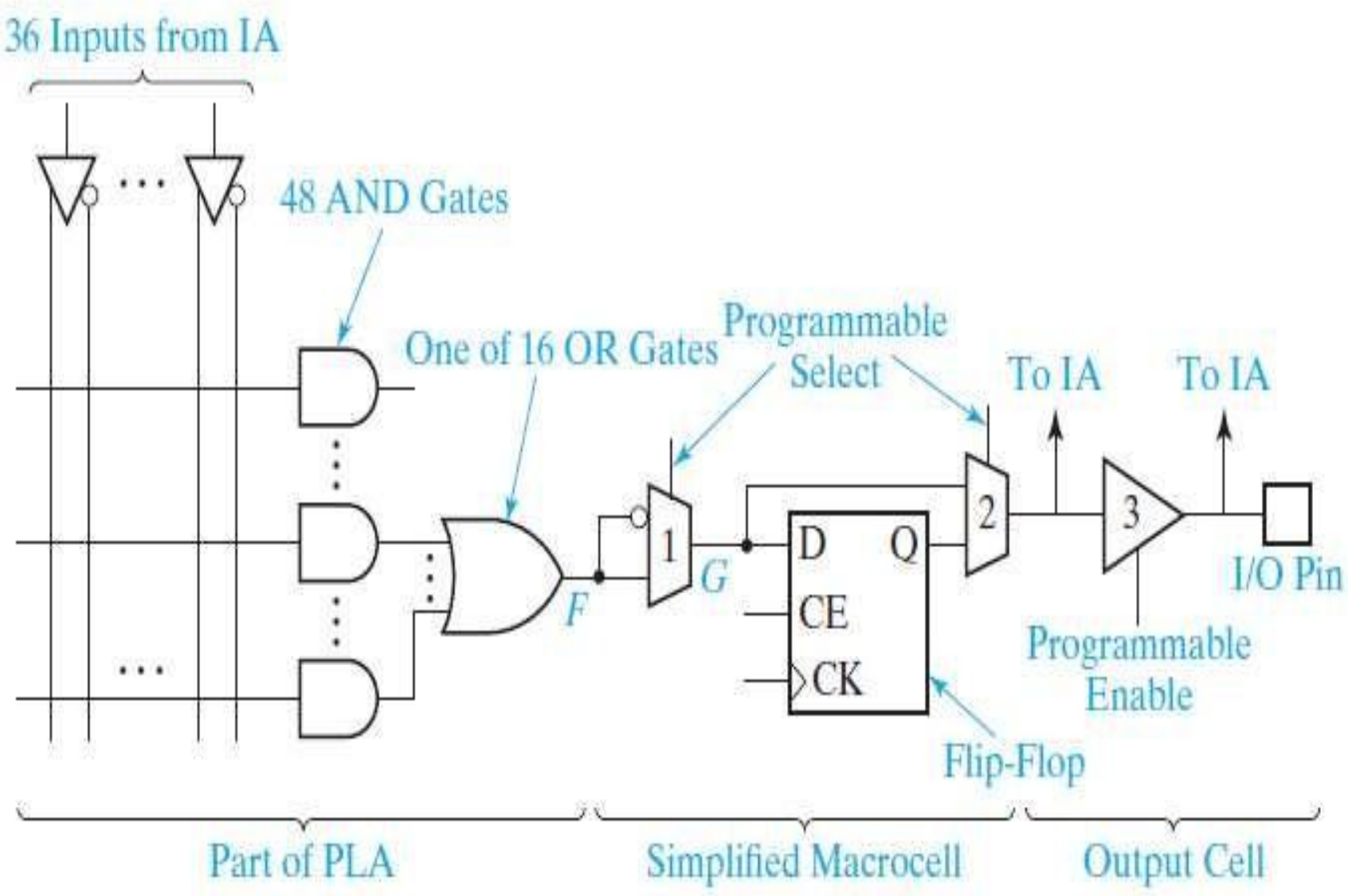
- As integrated circuit technology continues to improve, more and more gates can be placed on a single chip
- Instead of a single PAL or PLA on a chip, many PALs or PLAs can be placed on a single CPLD chip and interconnected.
- When storage elements such as flip-flops are also included on the same IC, a small digital system can be implemented with a single CPLD.
- Figure shows the basic architecture Xilinx XCR3064XL CPLD


I/O Pins
I/O
MC1
MC2
MC16
FUNCTION BLOCK
36
16
Interconnect Array (IA)

CONTD..

- Has four function blocks,
- Each block has 16 associated macrocells (MC1,MC2, .)
- Each function block is a programmable AND-OR array that is configured as a PLA.
- Each macrocell contains a flip-flop and multiplexers that route signals from the function block to the input-output (I/O) block or to the interconnect array (IA).
- The IA selects signals from the macrocell outputs or I/O blocks and connects them back to function block inputs.
- Thus, a signal generated in one function block can be used as an input to any other function block.
- The I/O blocks provide an interface between the bi-directional I/O pins on the IC and the interior of the CPLD

CPLD FUNCTION BLOCK AND MACROCELL



CONTD..

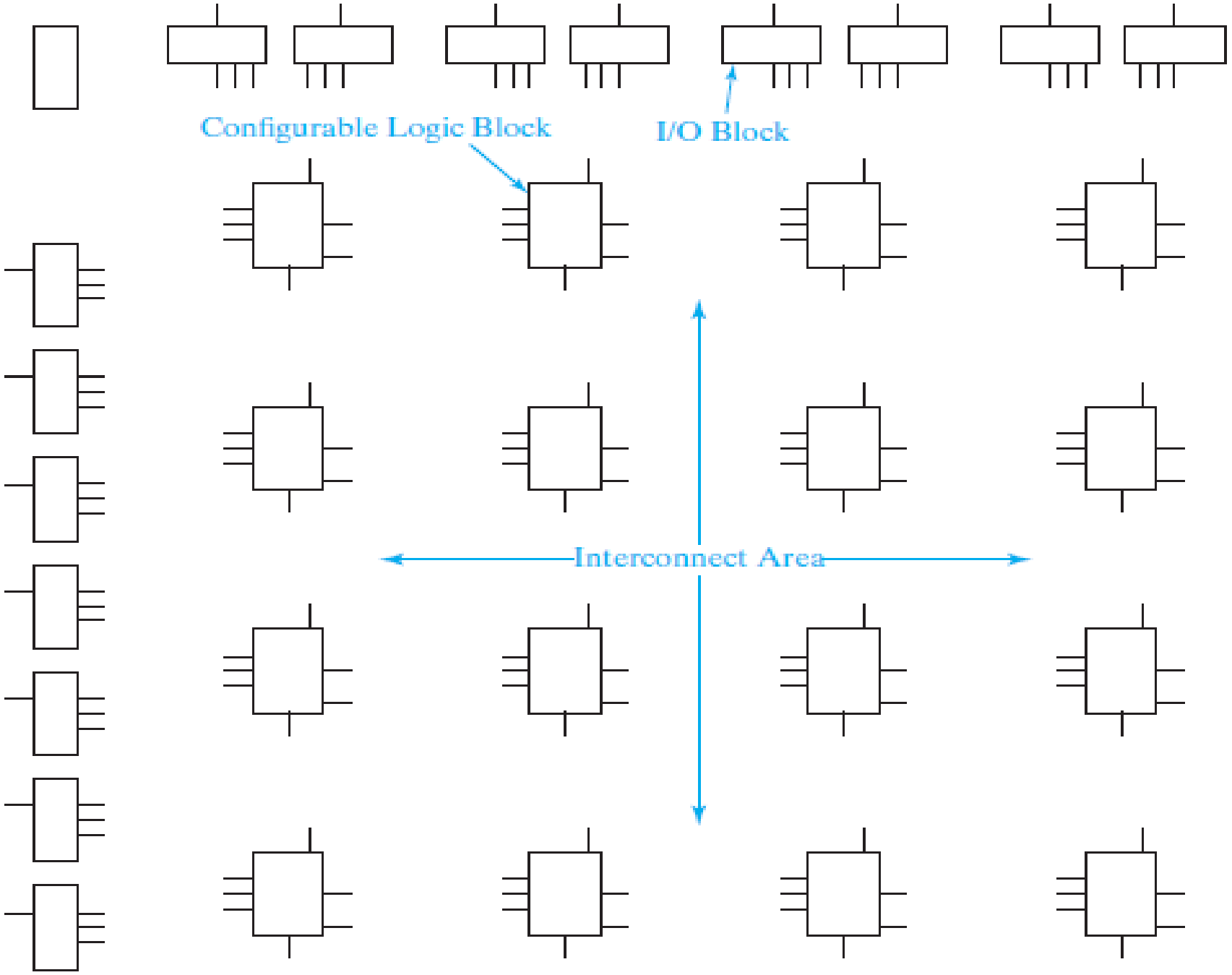
- Figure shows how a signal generated in the PLA is routed to an I/O pin through a macrocell.
- Any of the 36 outputs from the IA (or their complements) can be connected to any inputs of the 48 AND gates.
- Each OR gate can accept up to 48 product term inputs from the AND array.
- The first MUX (1) can be programmed to select the OR-gate output or its complement.
- The MUX (2) at the output of the macrocell can be programmed to select either the combinational output (*G) or the flip-flop output (Q)*.
- This output goes to the interconnect array and to the output cell.

CONTD..

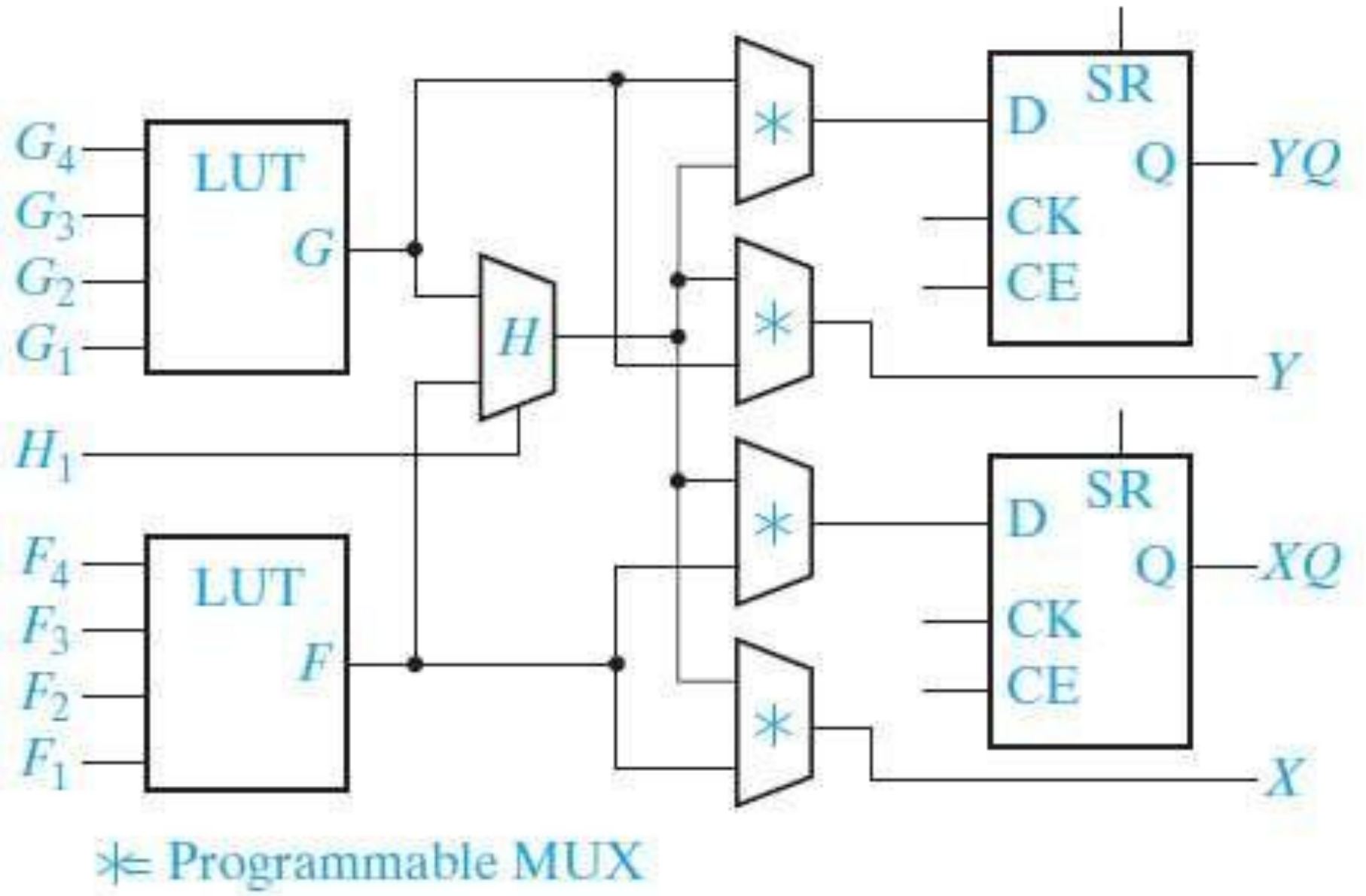
- The output cell includes a three-state buffer (3) to drive the I/O pin.
- When the I/O pin is used as an input, the buffer must be disabled

FIELD-PROGRAMMABLE GATE ARRAYS

- An FPGA is an IC that contains an array of identical logic cells with programmable interconnections.
- The user can program the functions realized by each logic cell and the connections between the cells.
- Figure shows the layout of part of a typical FPGA.
- The interior of the FPGA consists of an array of logic cells, also called configurable logic blocks (CLBs).
- The array of CLBs is surrounded by a ring of input-output interface blocks.
- These I/O blocks connect the CLB signals to IC pins.
- The space between the CLBs is used to route connections between the CLB outputs and inputs.


Configurable Logic Block
I/O Block
Interconnect Area

SIMPLIFIED CONFIGURABLE LOGIC BLOCK (CLB)



CONTD..

- This CLB contains two function generators, two flip-flops, and various multiplexers for routing signals within the CLB.
- Each function generator has four inputs and can implement any function of up
- to four variables. The function generators are implemented as lookup tables (LUTs).
- A four-input LUT is essentially a reprogrammable ROM with 16 1-bit words. This
- ROM stores the truth table for the function being generated. The *H multiplexer*
- selects either *F or G depending on the value of H1. The CLB has two combinational*
- outputs (*X and Y) and two flip-flop outputs (XQ and YQ). The X and Y outputs and*

DECOMPOSITION OF SWITCHING FUNCTIONS

- To implement a switching function of more than four variables using 4- variable function generators, the function must be decomposed into subfunctions.
- One method of decomposition is based on Shannon's expansion theorem.
- Expanding a function of the variables *a, b, c, and d about the variable a*:

$$f(a, b, c, d) = a' f(0, b, c, d) + a f(1, b, c, d)$$

$$= a' f0 + a f1$$

CONTD..

- The 3-variable function

  f0 = f(0, b, c, d) is formed by replacing a with 0 in f(a, b, c, d), and

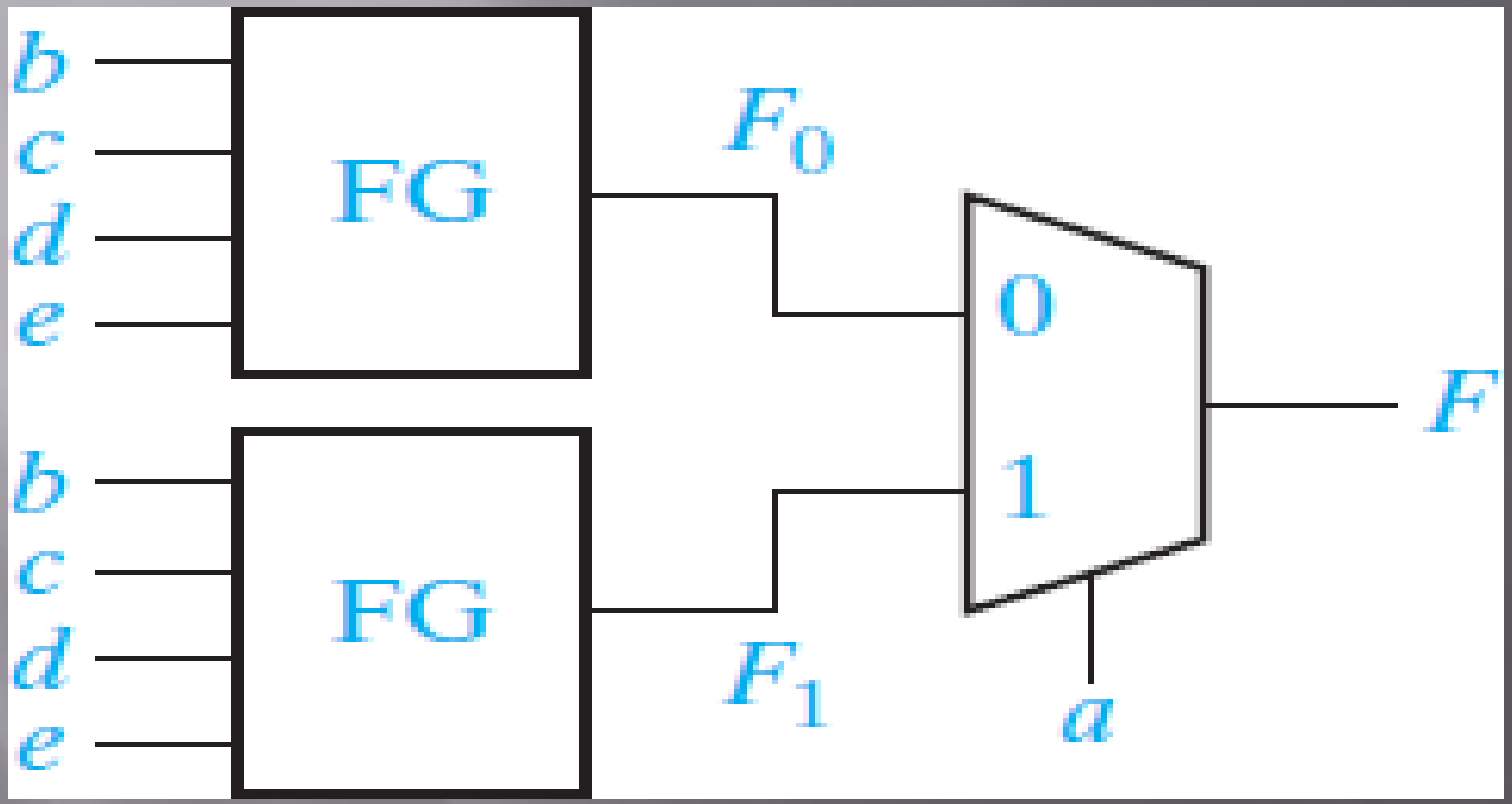
  f1 = f(1, b, c, d) is formed by replacing a with 1 in f(a, b, c, d).

- The expansion theorem to a 5-variable function gives

  $$f(a, b, c, d, e) = a' f(0, b, c, d, e) + a f(1, b, c, d, e)$$
  $$= a' f0 + a f1$$

- This shows that any 5-variable function can be realized using two 4-variable function generators and a 2-to-1 MUX

(a) 5-variable function

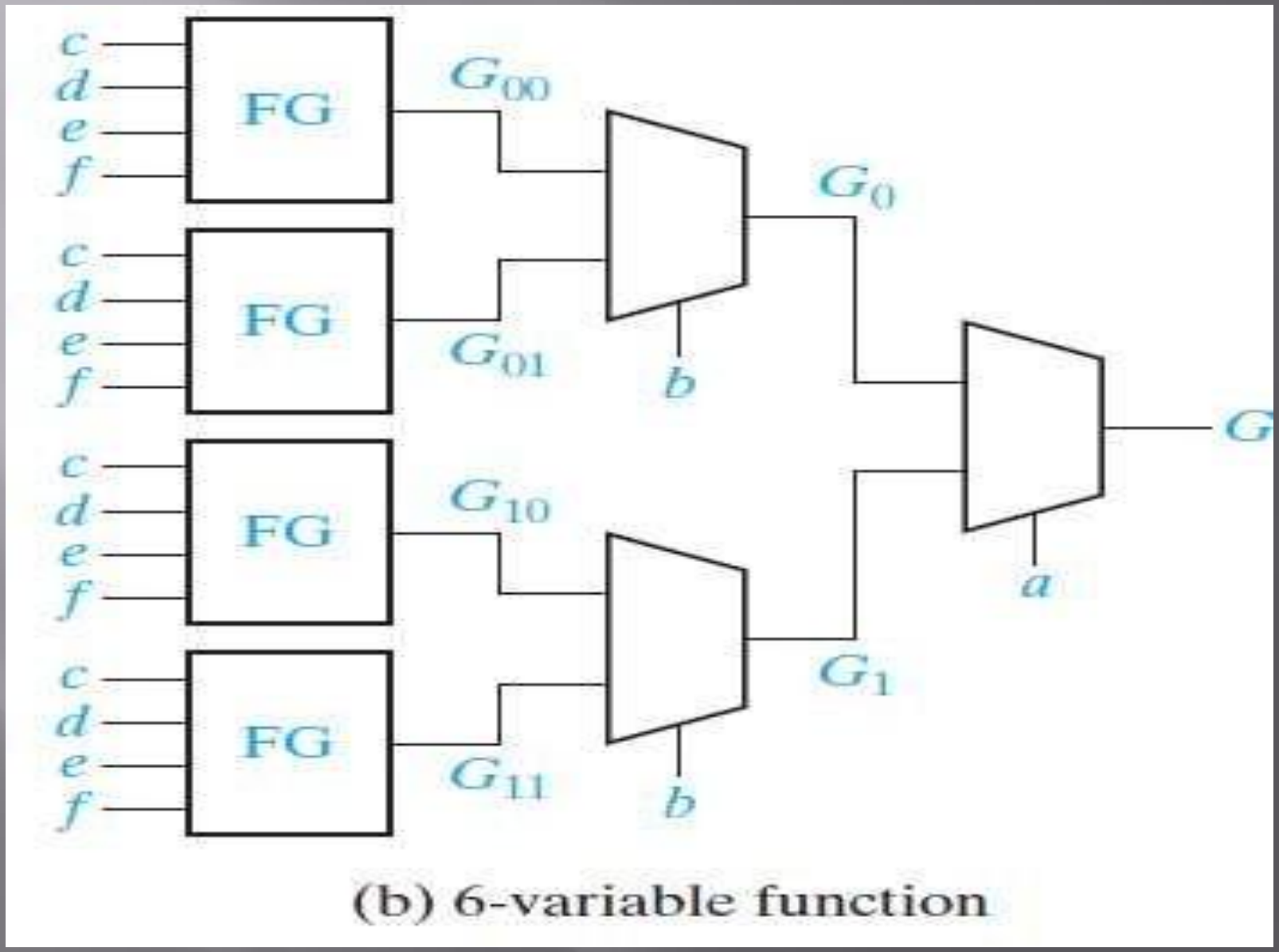


- To realize a 6-variable function using 4-variable function generators,the expansion theorem twice

$$G(a, b, c, d, e, f) = a'G(0, b, c, d, e, f) + a\,G(1, b, c, d, e, f)$$
$$= a'G0 + a\,G1$$
$$G0 = b'G(0, 0, c, d, e, f) + b\,G(0, 1, c, d, e, f)$$
$$= b'G00 + b\,G01$$
$$G1 = b'G(1, 0, c, d, e, f) + b\,G(1, 1, c, d, e, f)$$
$$= b'G10 + bG11$$

- Because *G00,G01,G10, and G11 are all 4-variable functions, we can realize any* 6-variable function using four 4-variable function generators and three 2-to-1 MUXes, as shown in Figure b
- Thus, we can realize any 6-variable function using two CLBs of the type shown in Figure

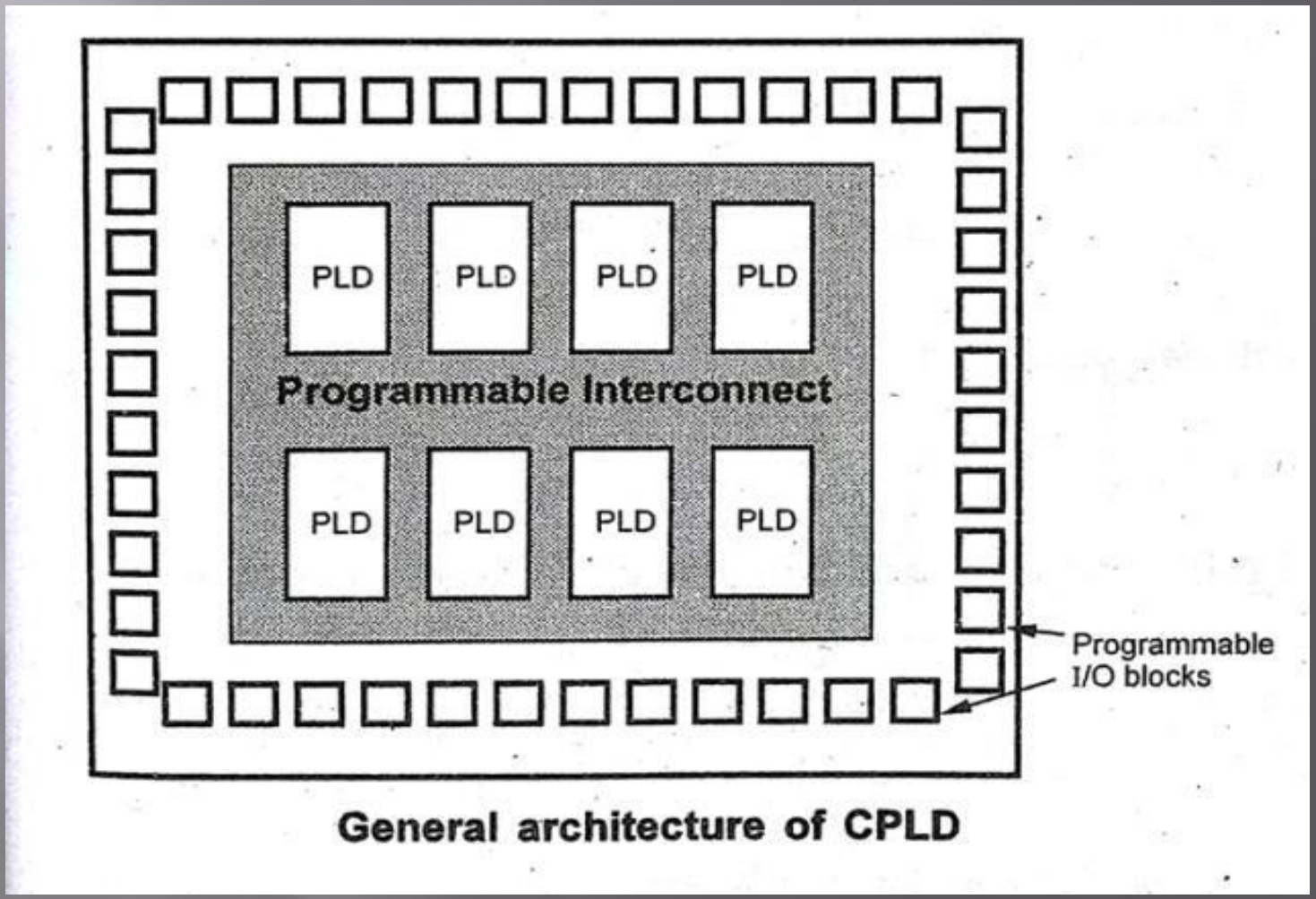
$$G(a, b, c, d, e, f) = a'b'G00 + a'b\,G01 + ab'G10 + ab\,G11$$



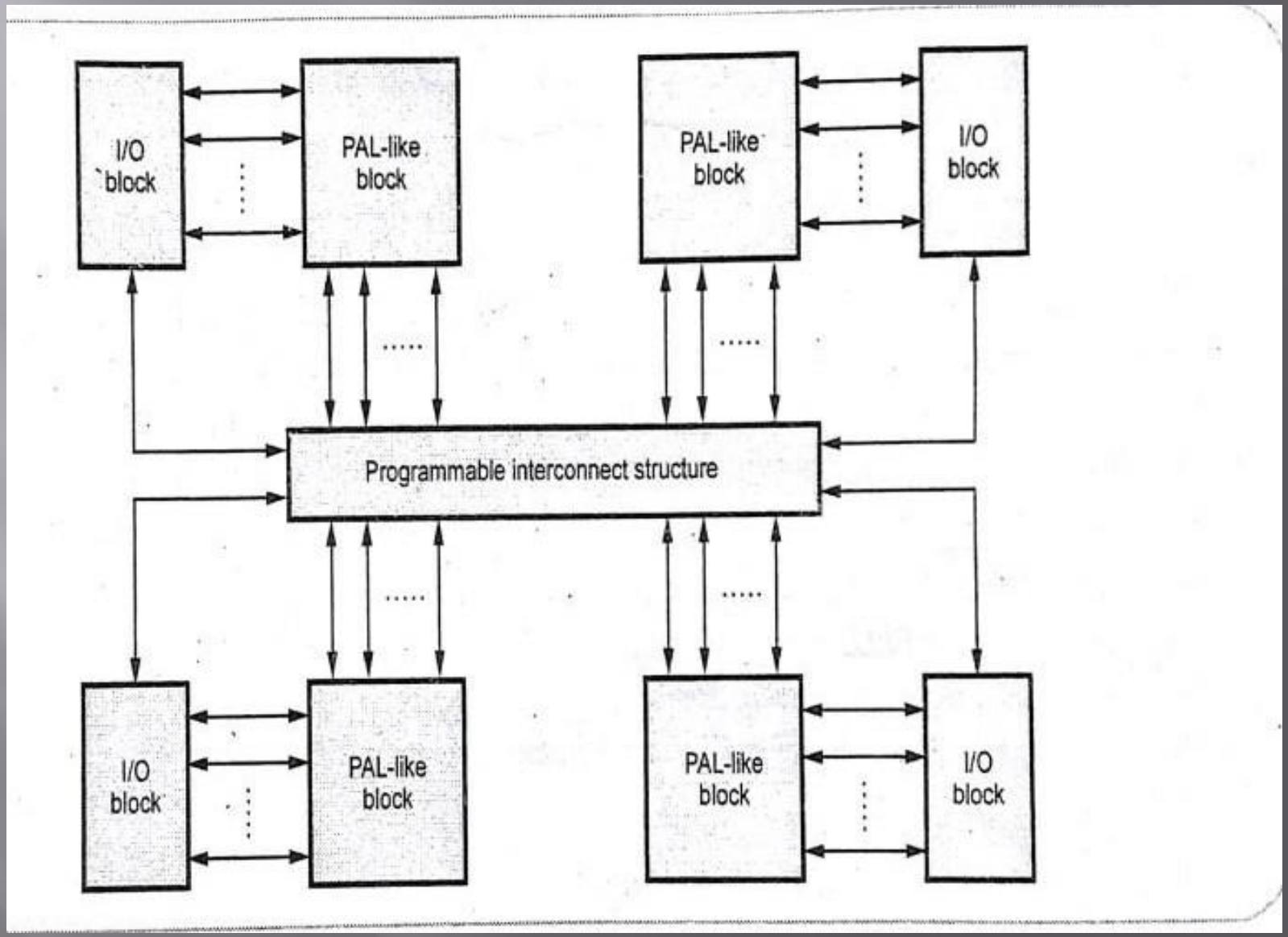
(b) 6-variable function

CPLD(Complex Programmable Logic Devices)

- It is merely a collection of multiple PLDs and an interconnection structure, all on the same chip.
- In addition to individual PLDs, the on-chip interconnection structure is also programmable.

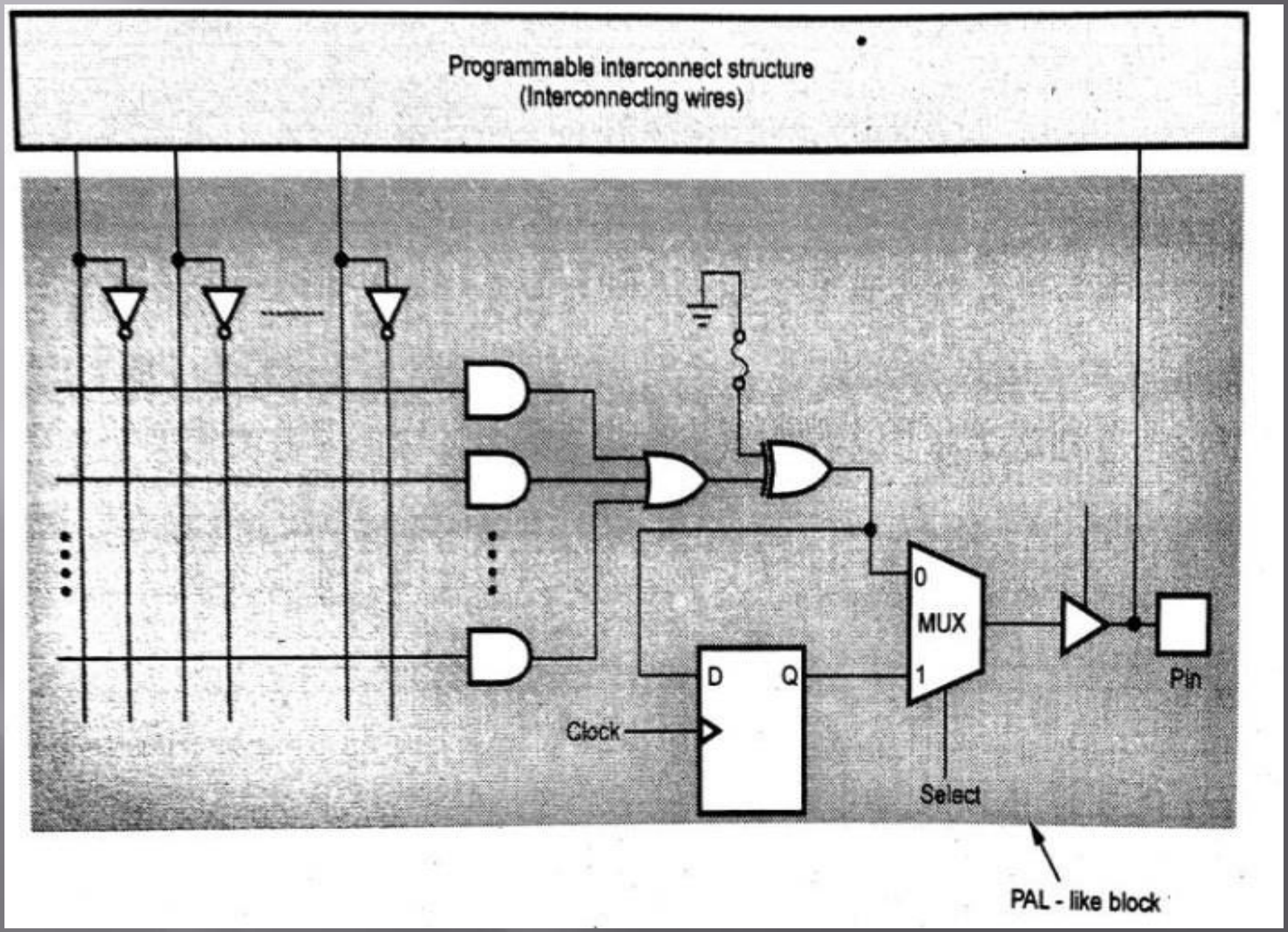


General architecture of CPLD


I/O block
PAL-like block
PAL-like block
I/O block
Programmable interconnect structure
I/O block
PAL-like block
PAL-like block
I/O block

Contd…

- It consists of collection of PAL like blocks, I/O blocks and a set of interconnection wires, called programmable interconnection structure.
- The PAL like blocks are connected to the programmable interconnect structure and to the I/O blocks and it usually consists of 16 macrocells.
- The macrocells in CPLD consists of AND-OR configuration, an EX-OR gate, a flip-flop, a multiplexer and a tri-state buffer.
- The EX-OR gate provides the output of OR-gate in inverted or non-inverted form as per the fuse link status.
- A D flip flop stores the output of EX-OR gate.
- Multiplexer selects either the output of the D flip flop or the output of the EX-OR gate depending on the select input.
- The tri-state buffer acts as a switch which enables or disables the output.

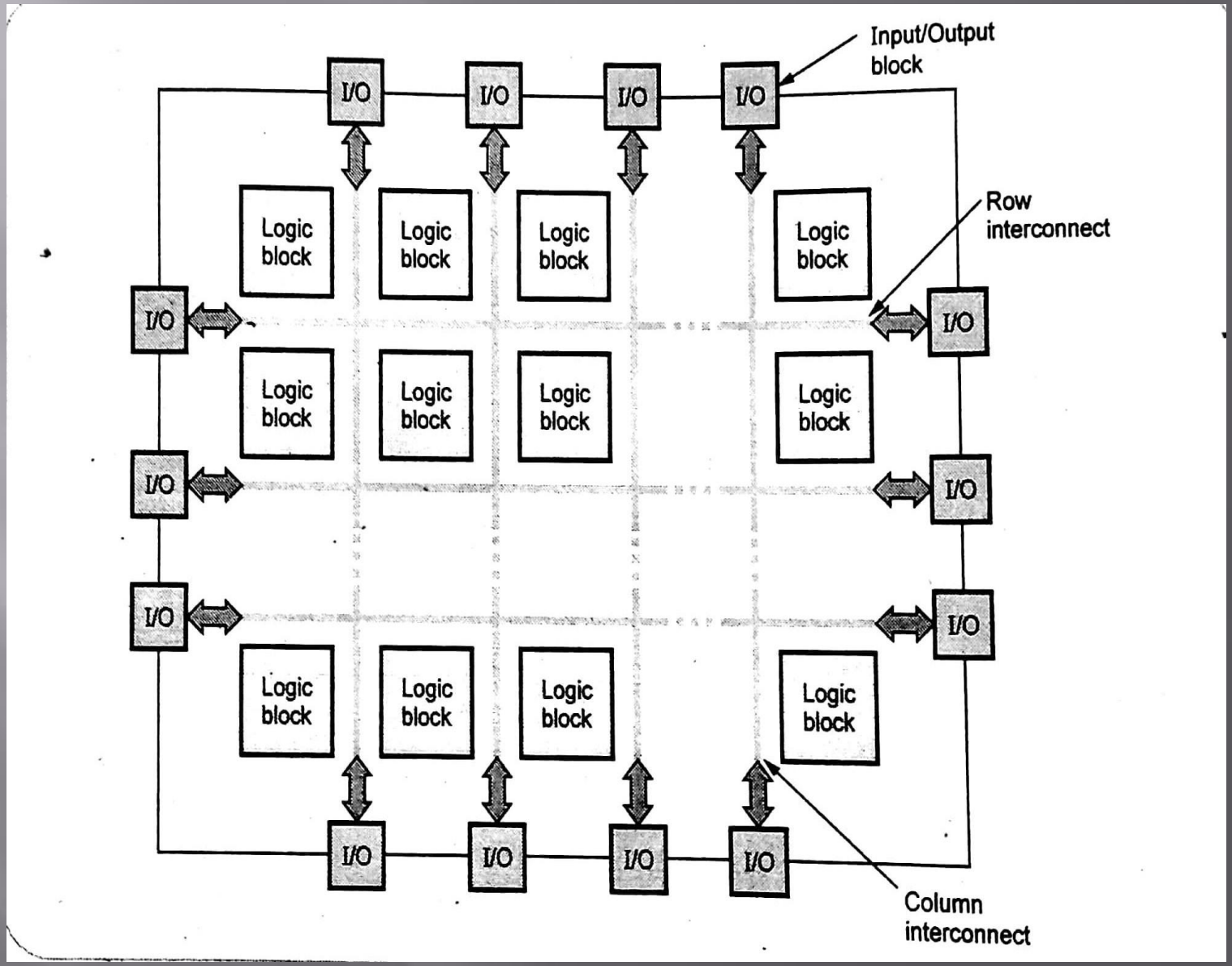

Programmable interconnect structure
(Interconnecting wires)
0
MUX
1
D
Q
Clock
Select
Pin
PAL - like block

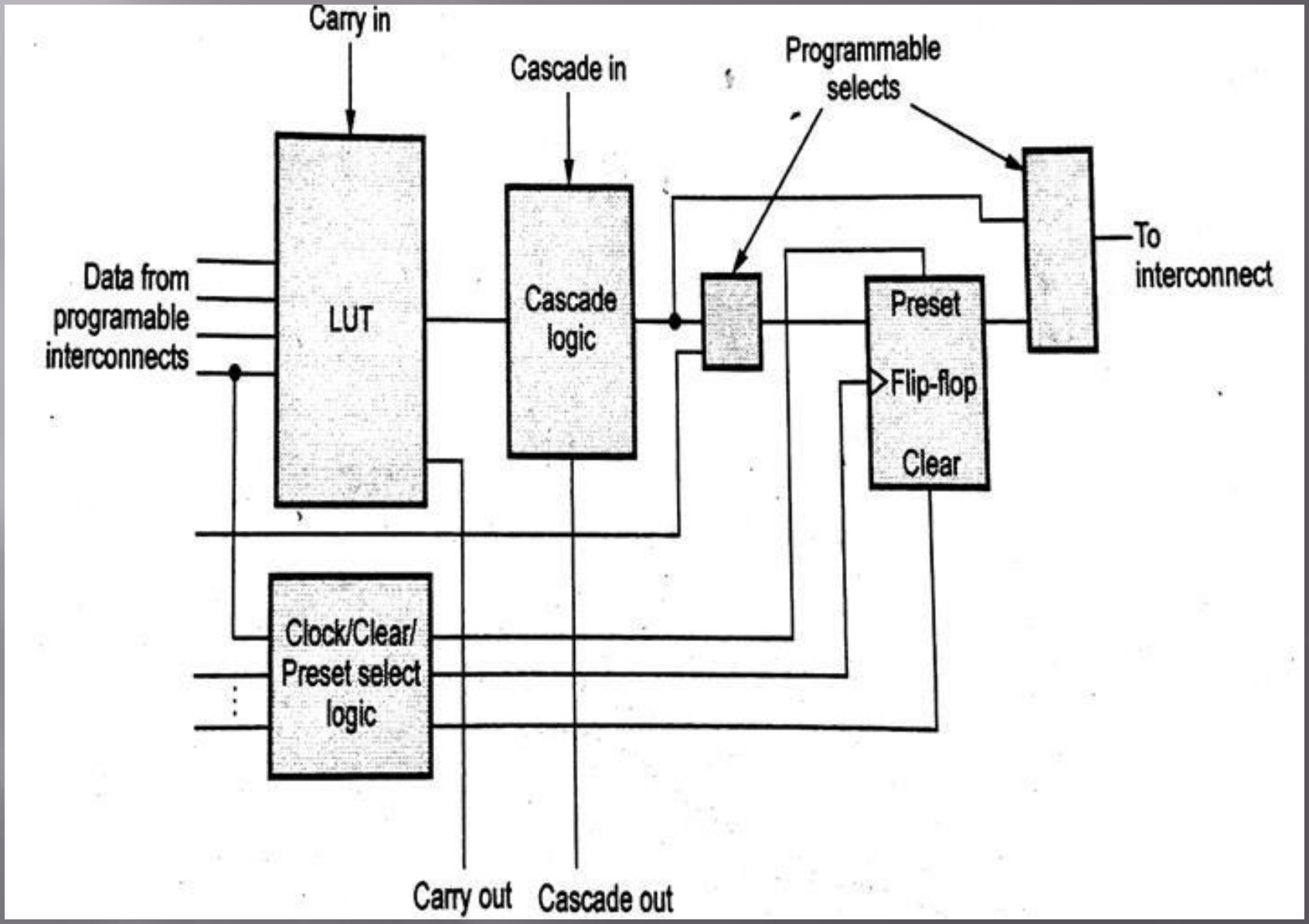
FPGA(Field Programmable Gate Arrays)

- The word field refers to the ability of the gate arrays to be programmed for the specific function by the user instead of by the manufacturer of the device.
- The word array is used to indicate the series of columns and rows of gates that can be programmed by the end user.
- The programmable logic blocks of FPGAs are called logic blocks or configurable logic blocks(CLBs).

Contd…

- The basic structure of FPGA consists of an array of logic blocks with programmable row and column interconnecting channels surrounded by programmable I/O blocks.
- Look-up table(LUT) is used as a memory device that can be programmed to perform logic functions.
- Each logic block in a generic FPGA contains several logic elements.
- The logic element consists of LUT, associated logic and a flip-flop.


Input/Output block
I/O
Logic block
Row interconnect
Column interconnect


Carry in
Cascade in
Programmable selects
To interconnect
Data from programable interconnects
LUT
Cascade logic
Preset
Flip-flop
Clear
Clock/Clear/ Preset select logic
Carry out
Cascade out

Difference between FPGA and CPLD

FPGA	CPLD
Consists of configurable logic blocks, I/O blocks, row interconnect and column interconnect.	Consists of PAL like blocks, I/O blocks and programmable interconnect structures.
They use memory called LUT or multiplexers to generate logic functions.	They use AND/OR arrays to generate logic functions.
They are programmed for specific function by the user.	They are programmed for specific function by the manufacture of the device.
More complex architecture.	Less complex architecture.
Delays are quite unpredictable.	Delays are more predictable.
It is more expensive	It is much cheaper
FPGAs are made up of tiny logic blocks.	CPLDs are made of larger blocks.

Conclusion:

The architecture of CPLD and FPGA has been discussed and their comparison has been listed.

References:

- Mandal, "Digital Electronics Principles & Application, McGraw Hill Edu, 2013.
- William Keitz, Digital Electronics-A Practical Approach with VHDL, Pearson, 2013.
- Thomas L.Floyd, 'Digital Fundamentals', 11th edition, Pearson Education, 2015.
- Charles H.Roth, Jr, Lizy Lizy Kurian John, 'Digital System Design using VHDL, Cengage,2013.
- D.P.Kothari,J.S.Dhillon, 'Digital circuits and Design',Pearson Education, 2016.
- A.P.Godse., Dr.D.A.Godse, 'Digital Logic Circuits' , Technical Publications Third Edition 2016
- Other Web Sources