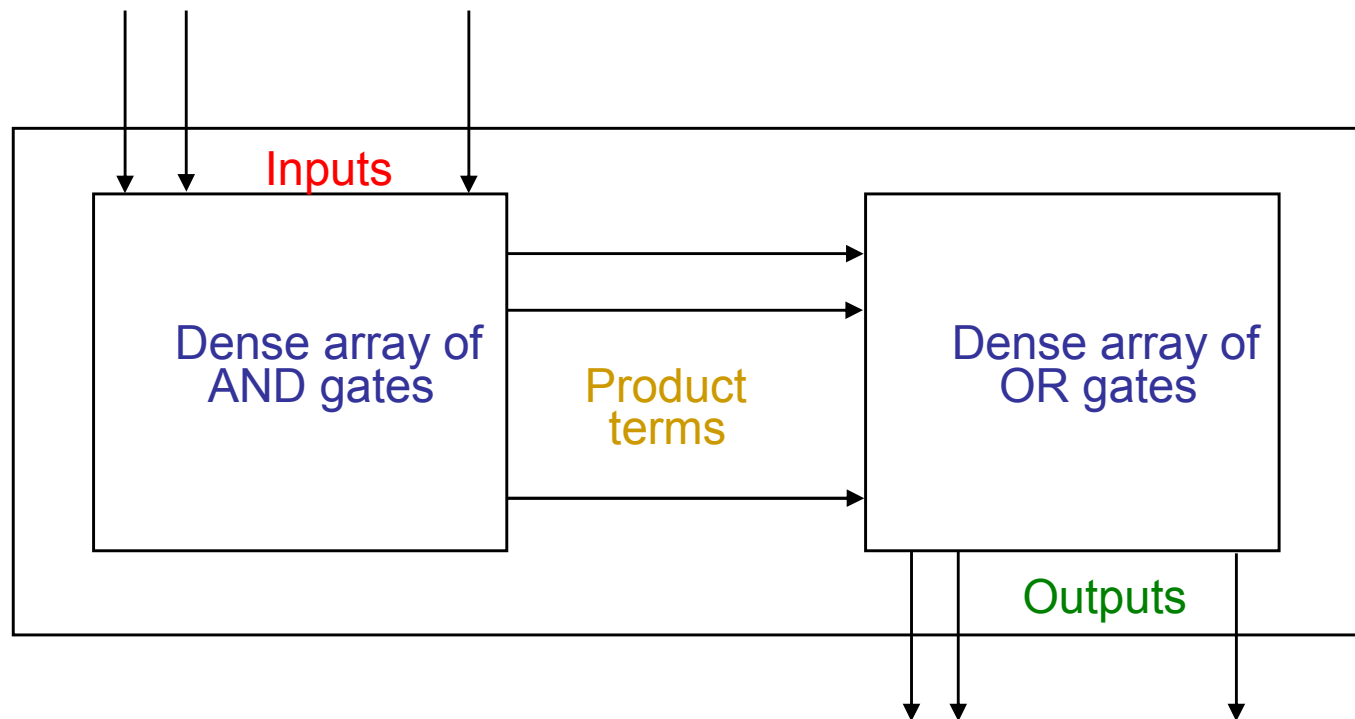


Programmable Logic

Programmable 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.

PLA Logic Implementation

Key to Success: Shared Product Terms

Equations

Example:

$$F_0 = A + \bar{B} \bar{C}$$

$$F_1 = \bar{A} \bar{C} + A B$$

$$F_2 = \bar{B} \bar{C} + A B$$

$$F_3 = \bar{B} C + A$$

Personality Matrix

Product term	Inputs			Outputs			
	A	B	C	F ₀	F ₁	F ₂	F ₃
A B	1	1	-	0	①	①	0
$\bar{B} \bar{C}$	-	0	1	0	0	0	1
A \bar{C}	1	-	0	0	1	0	0
$\bar{B} C$	-	0	0	①	0	①	0
A	1	-	-	①	0	0	①

Reuse of terms

Input Side:

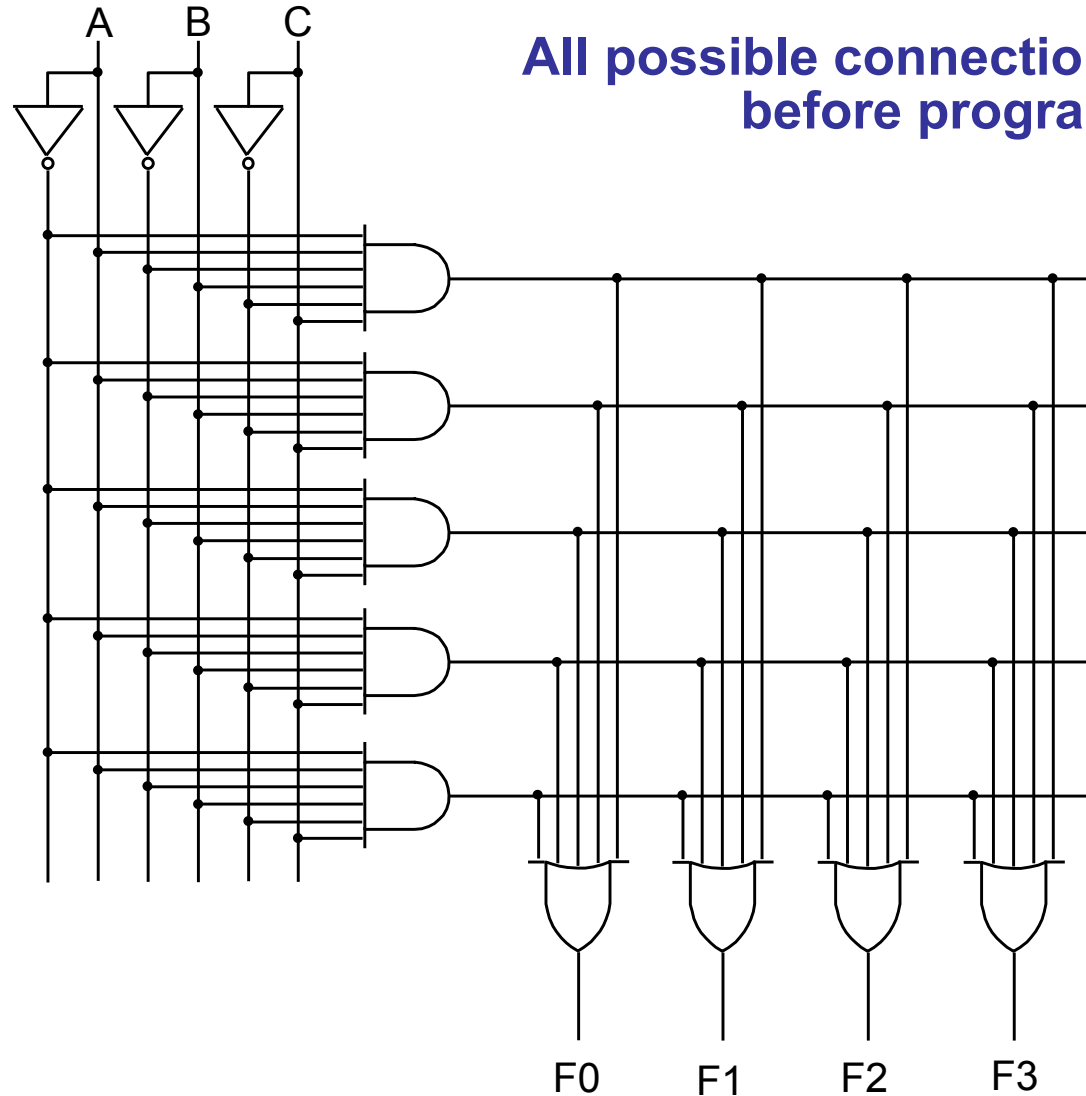
1 = asserted in term
 0 = negated in term
 - = does not participate

Output Side:

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

PLA Logic Implementation

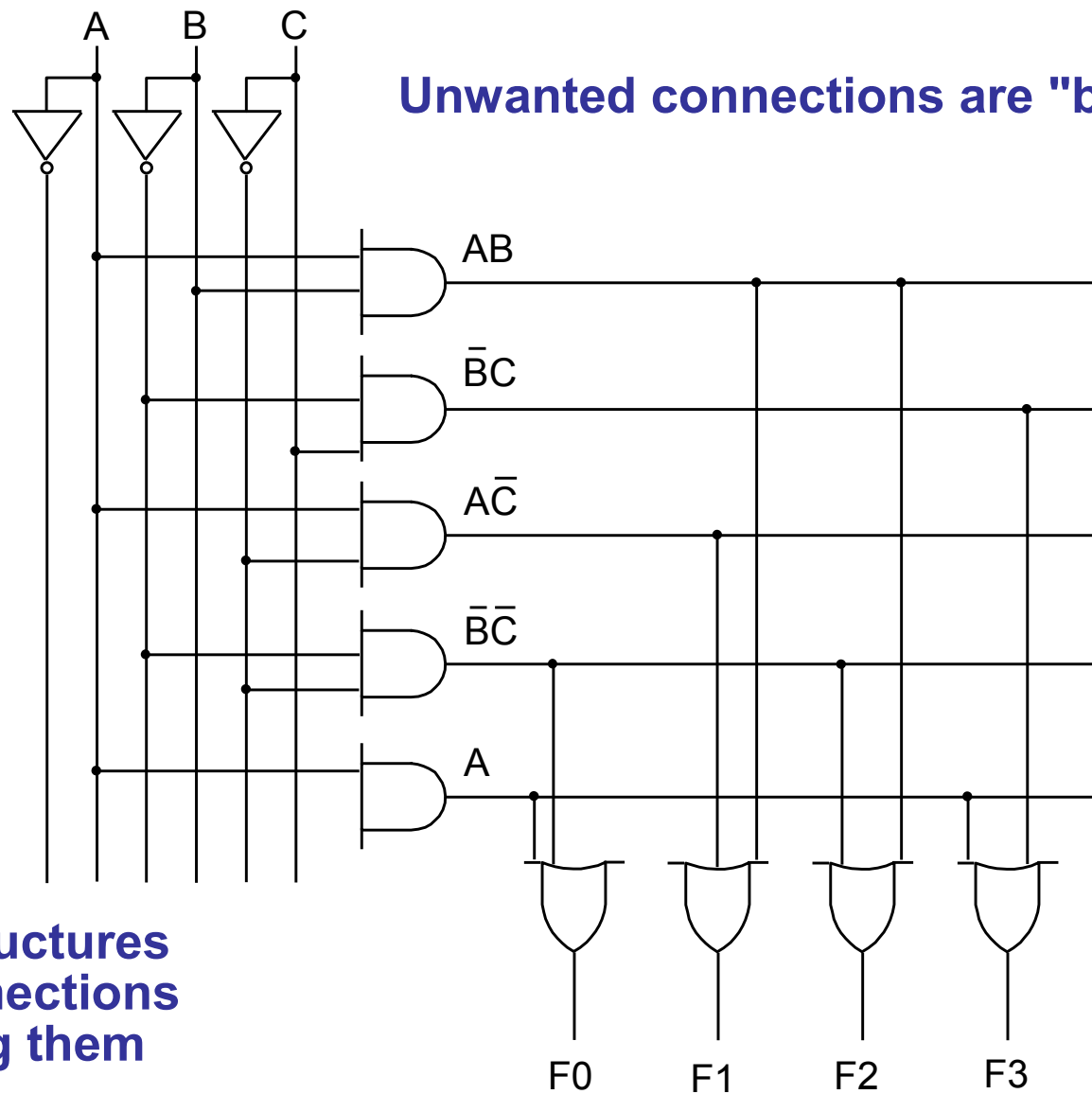
Example Continued – Un-programmed device



All possible connections are available before programming

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

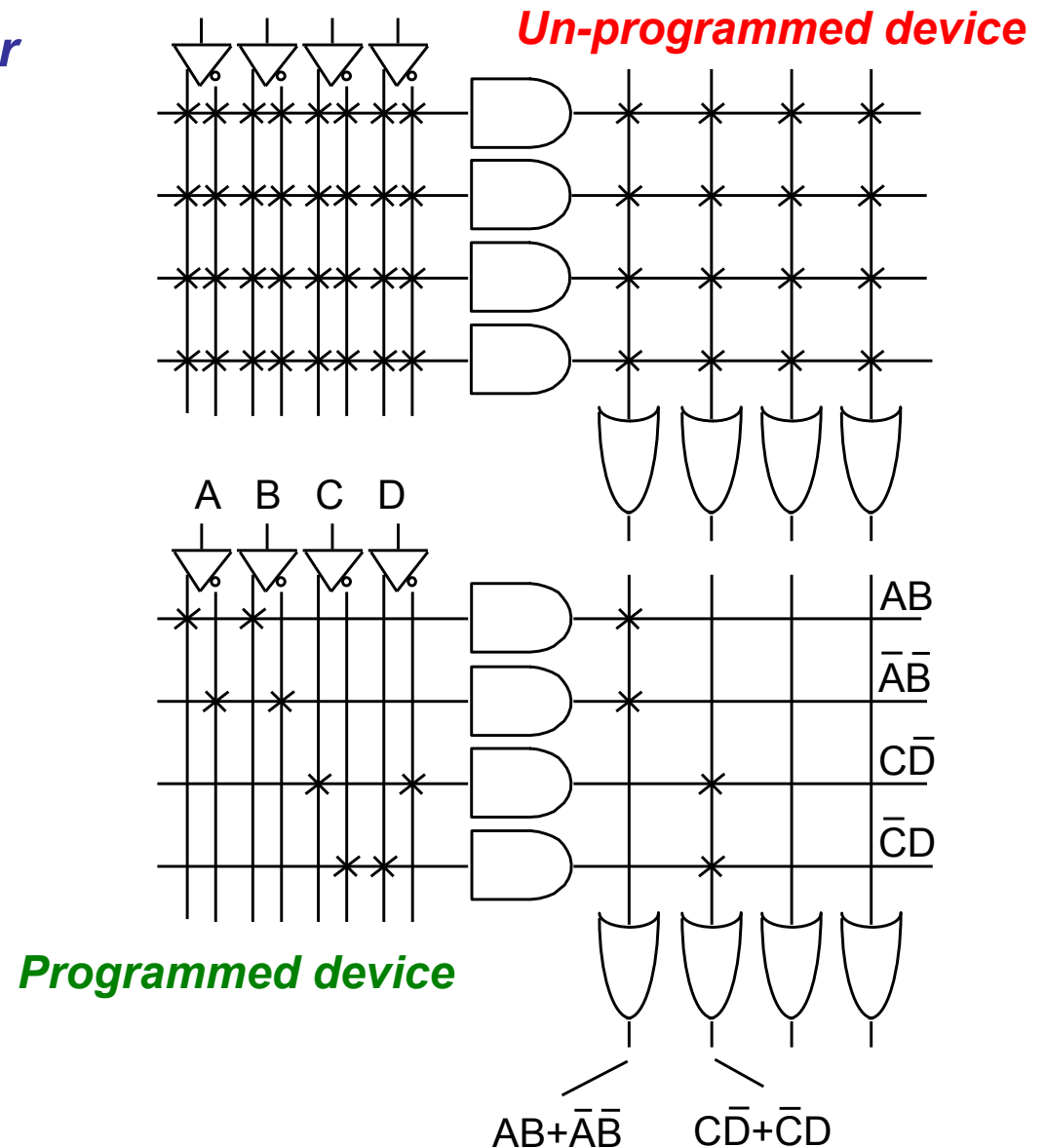
Short-hand notation so we don't have to draw all the wires!

X at junction indicates a connection

Notation for implementing

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

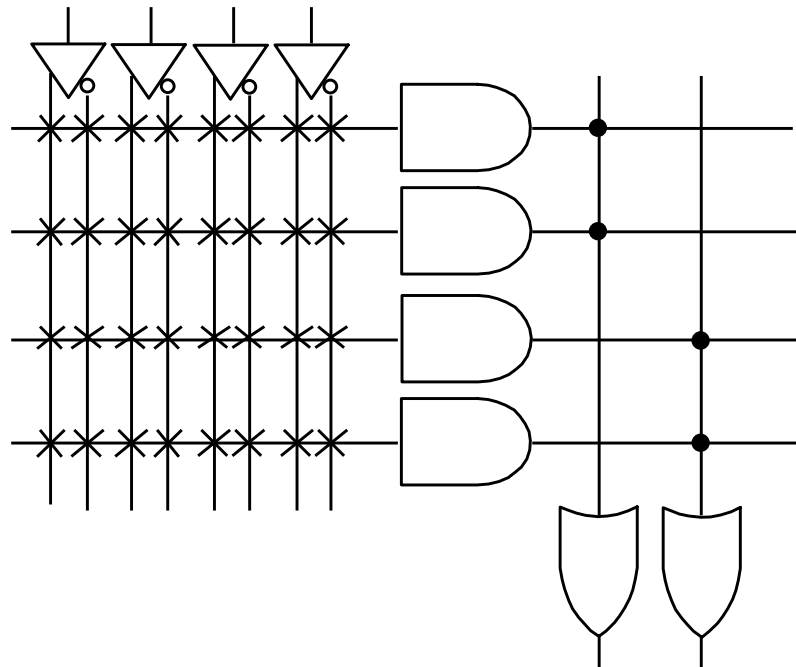
$$F1 = C \bar{D} + \bar{C} D$$



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

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

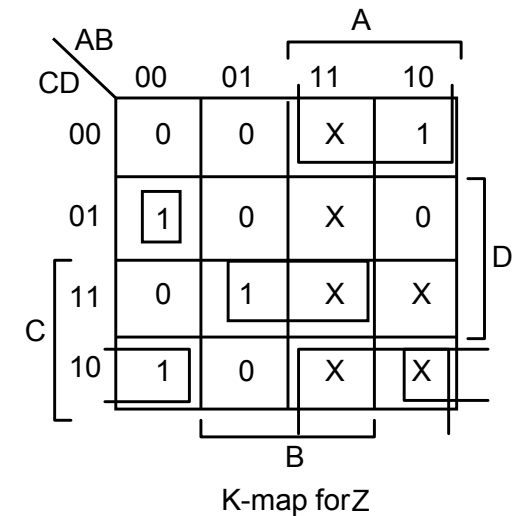
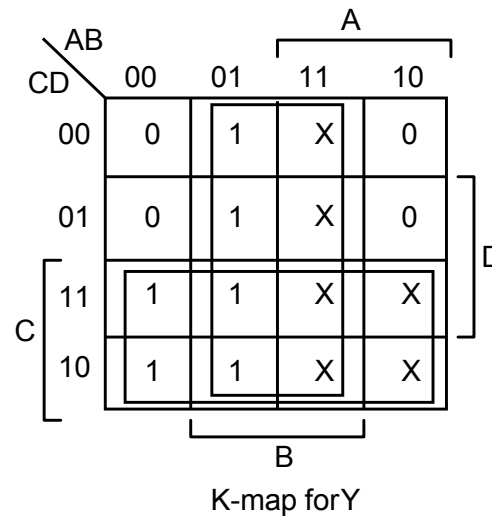
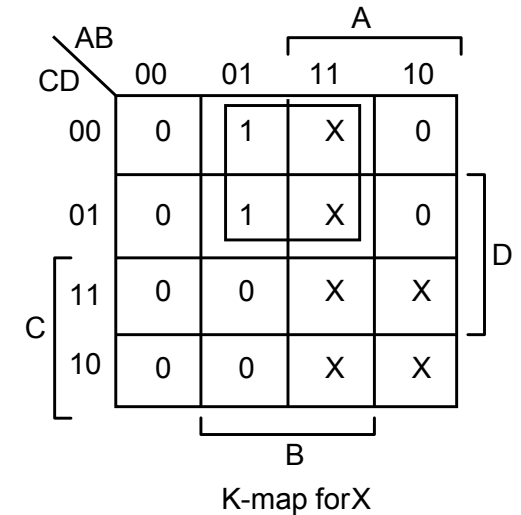
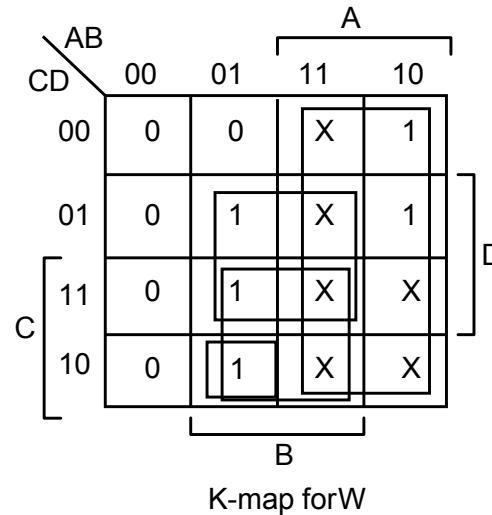
PAL Logic Implementation

Design Example: BCD to Gray Code Converter

K-maps

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



Minimized Functions:

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

$$X = B \bar{C}$$

$$Y = B + C$$

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

PAL Logic Implementation

Programmed PAL:

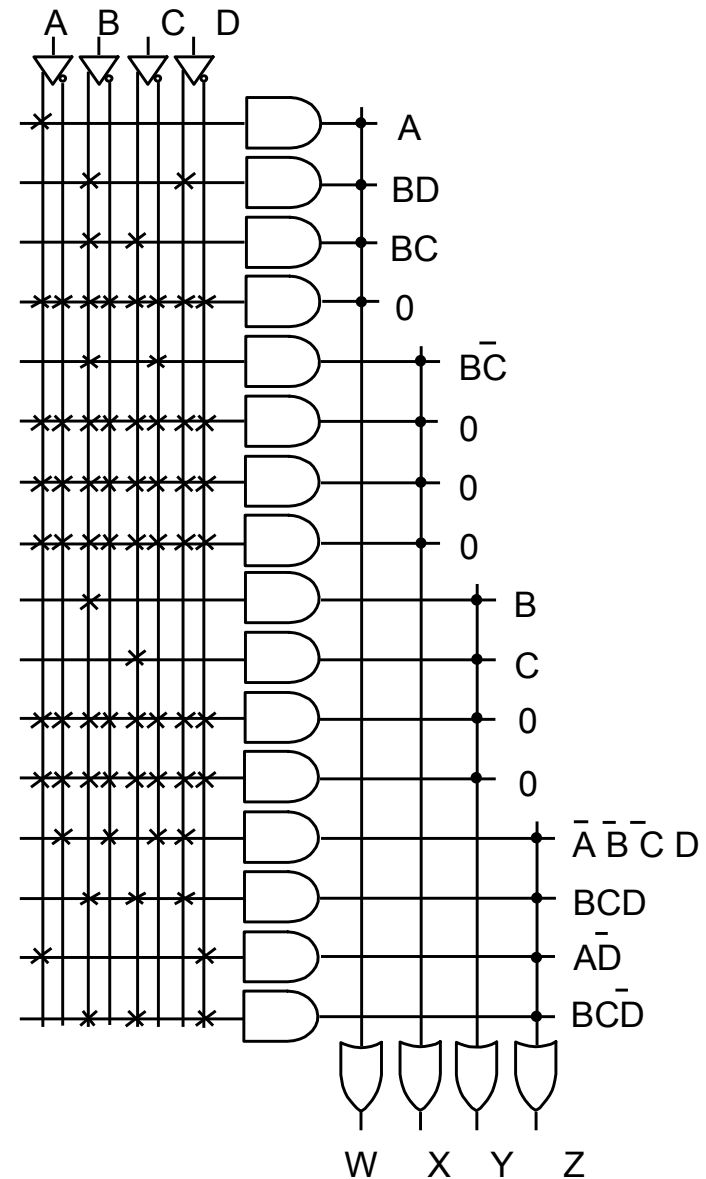
Minimized Functions:

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

$$X = B \bar{C}$$

$$Y = B + C$$

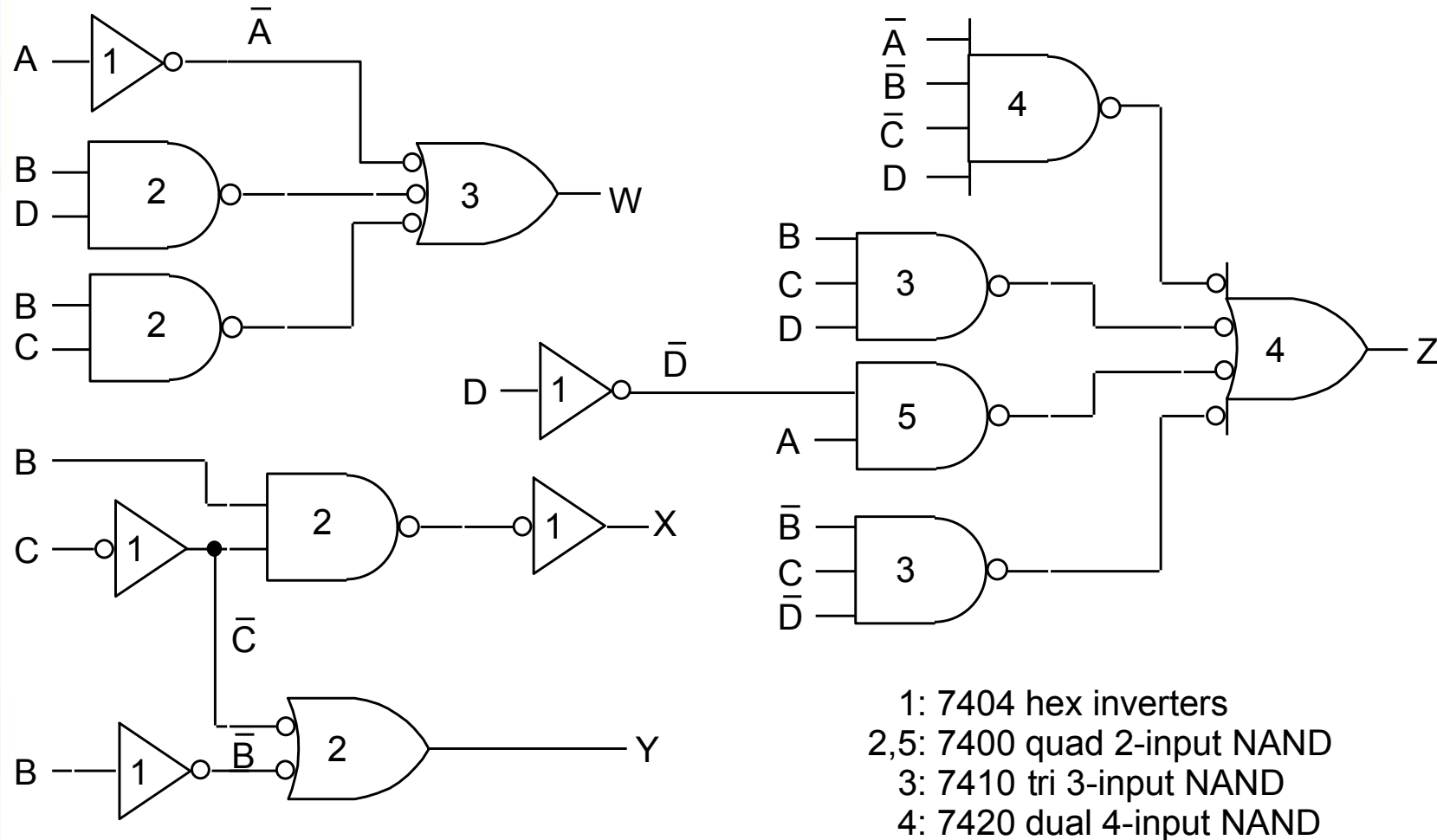
$$Z = \bar{A} \bar{B} \bar{C} D + B C D + A \bar{D} + \bar{B} C \bar{D}$$



4 product terms per each OR gate

PAL Logic Implementation

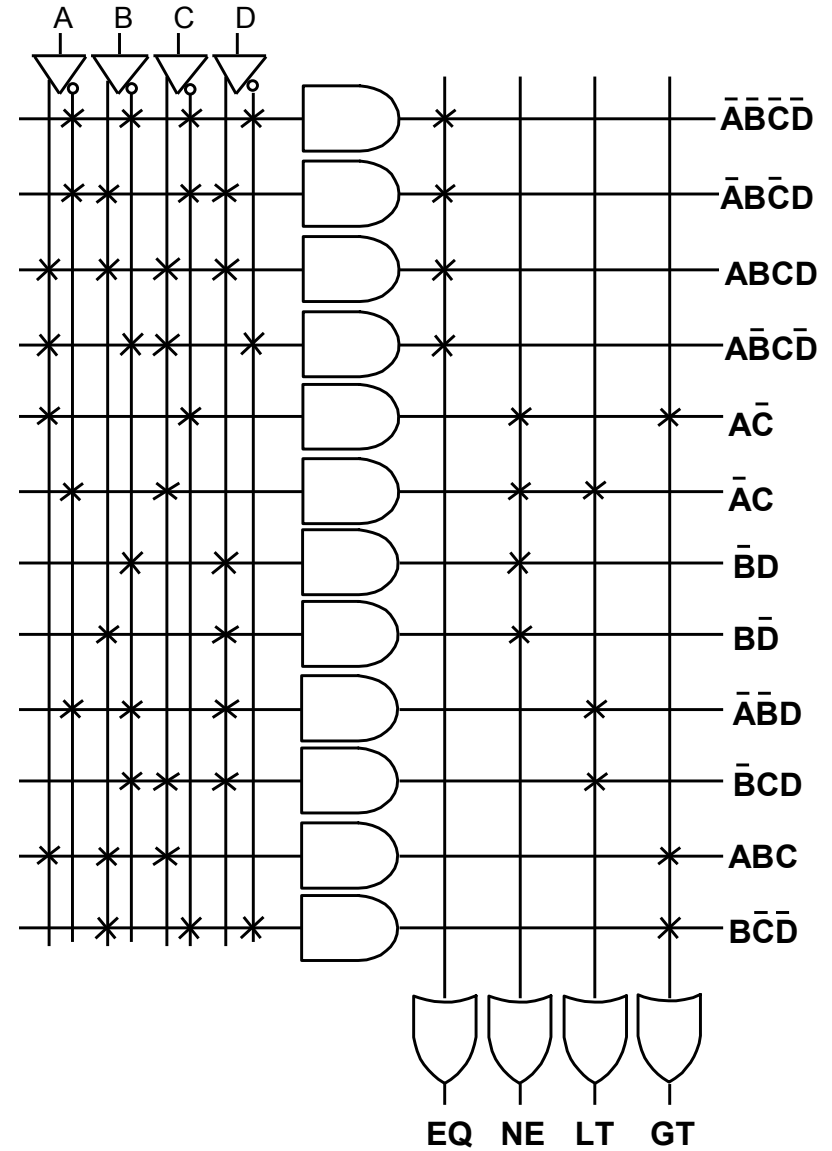
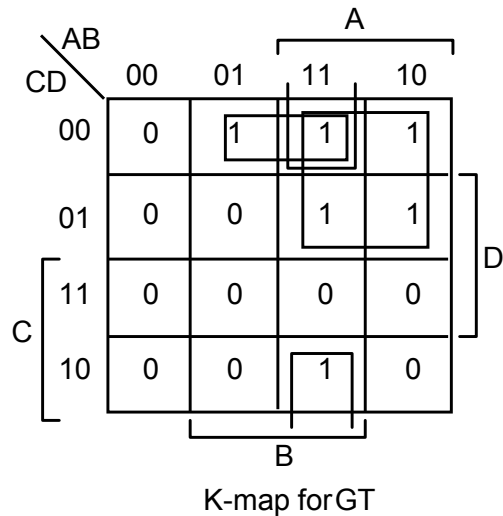
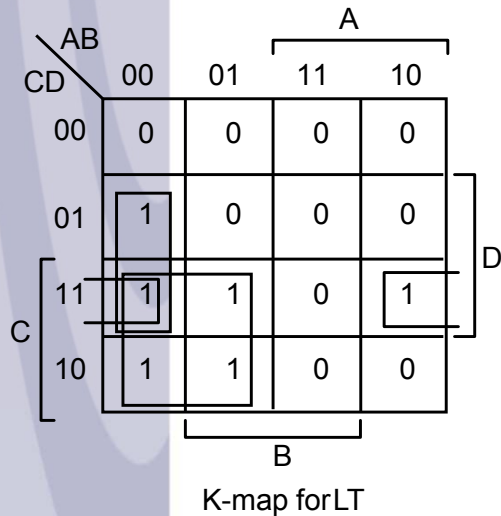
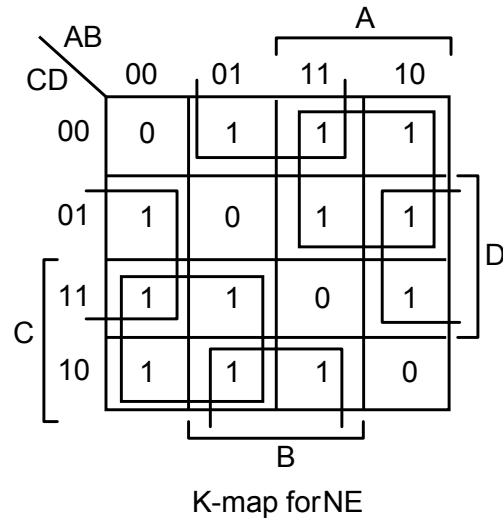
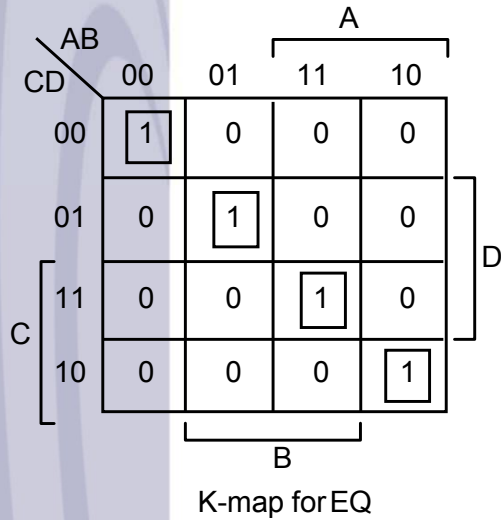
Code Converter Discrete Gate Implementation



4 SSI Packages vs. 1 PLA/PAL Package!

PLA Logic Implementation

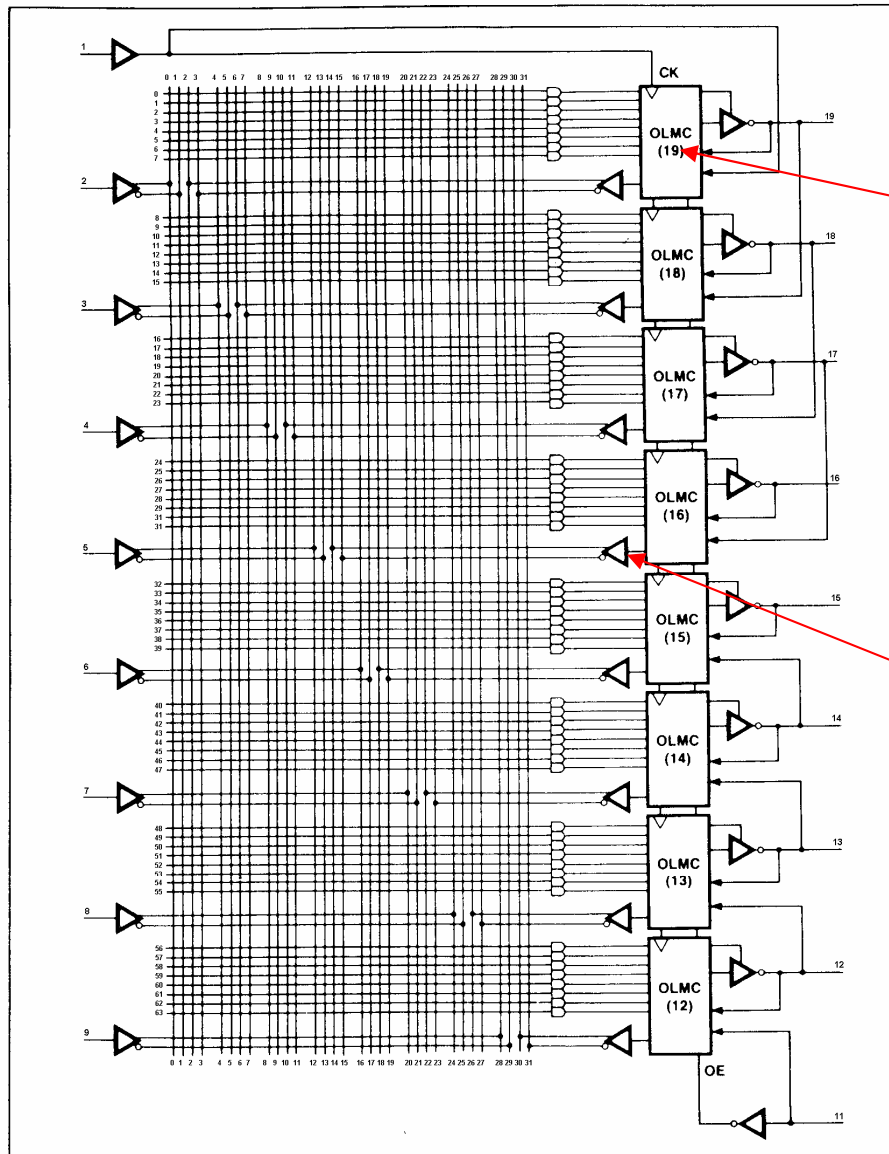
Another Example: Magnitude Comparator AB Vs. CD



Complex Programmable Logic Devices

- Complex PLDs typically combine PAL combinational logic with FFs
 - Organized into logic blocks
 - Fixed OR array size
 - Combinational or registered output
 - Some pins are inputs only
- Usually enough logic for simple counters, state machines, decoders, etc.
- e.g. 22G10, 20V8, etc.

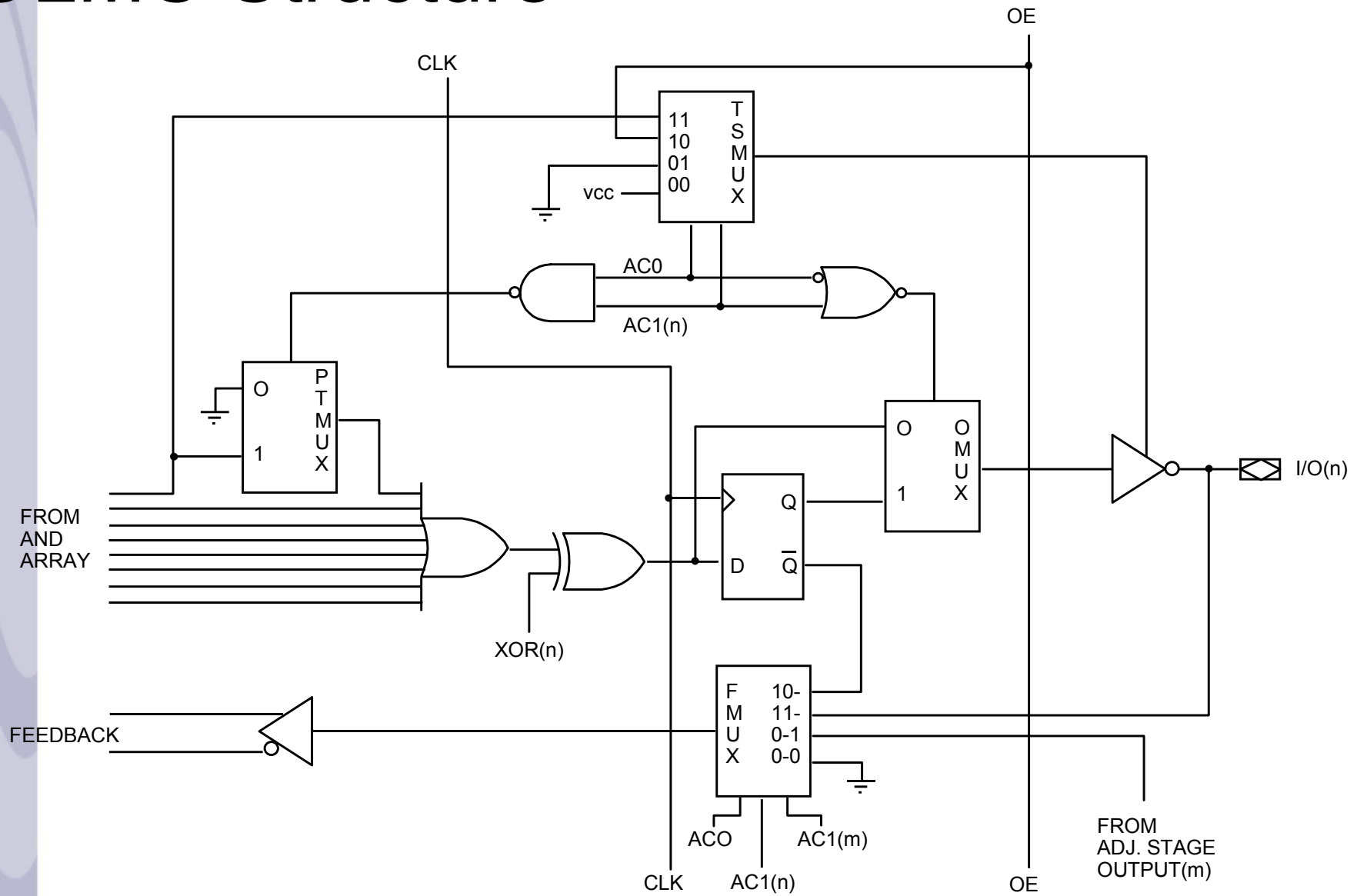
16V8 CPLD



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.

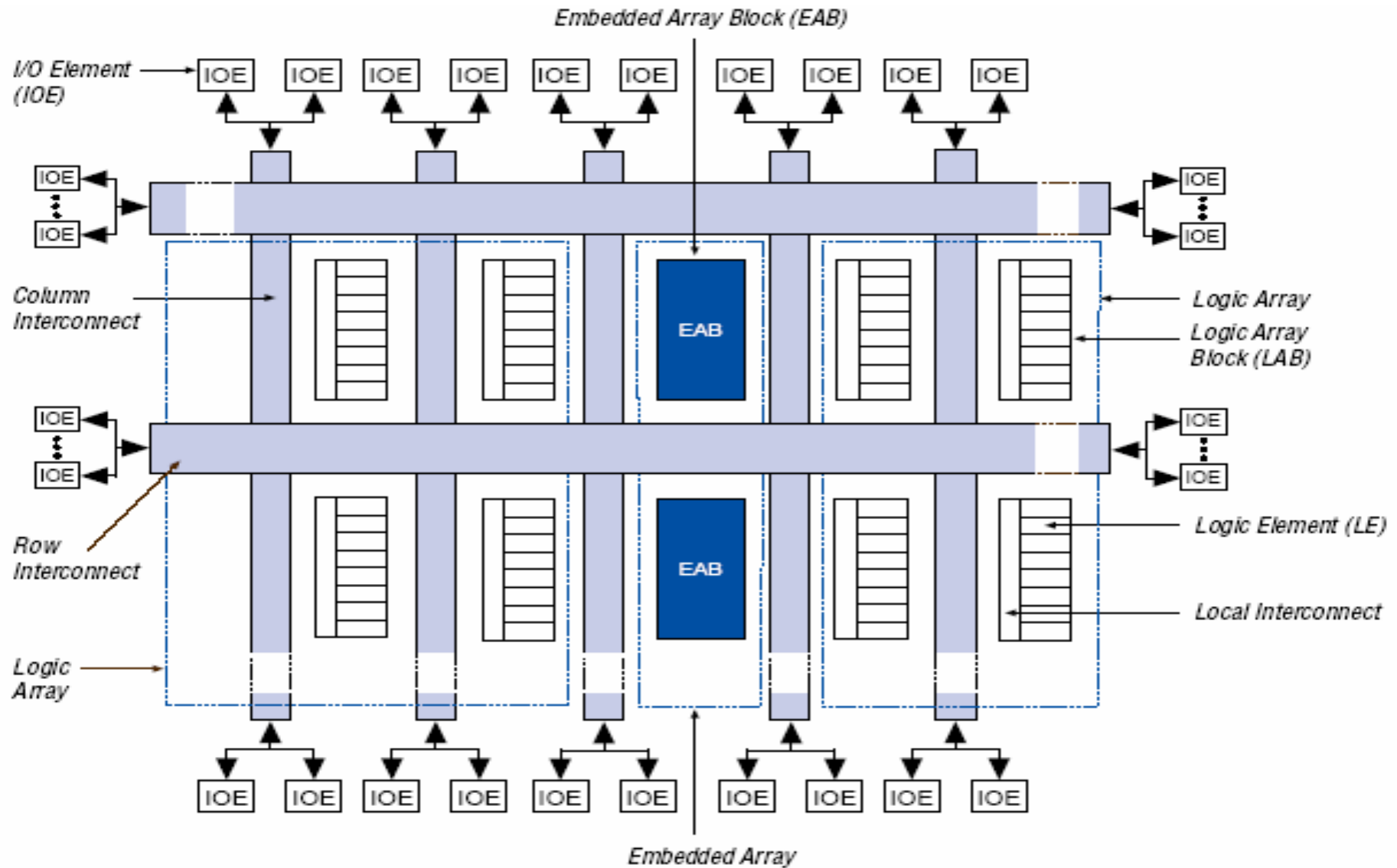
OLMC Structure



Field Programmable Gate Arrays (FPGAs)

- FPGAs have much more logic than CPLDs
 - ❖ 2K to >10M equivalent gates
 - ❖ Requires different architecture
 - ❖ FPGAs can be RAM-based or Flash-based
 - RAM FPGAs must be programmed at power-on
 - External memory needed for programming data
 - May be dynamically reconfigured
 - Flash FPGAs store program data in non-volatile memory
 - Reprogramming is more difficult
 - Holds configuration when power is off
- Logic Implemented by LUTs

Altera Flex 10k internal architecture



Altera *Flex 10k* internal architecture

- Typical organization

- ❖ LE – Logic Element contain functional logic.

- Combinational functions plus FFs

- Complexity varies by device

Note – in new devices called ALM – Adaptive Logic Module

- ❖ LAB - Logic Array Block contains 8 LEs.

- ❖ LA – Logic Array, A unit of 4 LABs

- ❖ EAB – Embedded Array Block, Implementation of Memory units :
RAM, ROM, FIFO...

- ❖ Interconnection is either local or long line

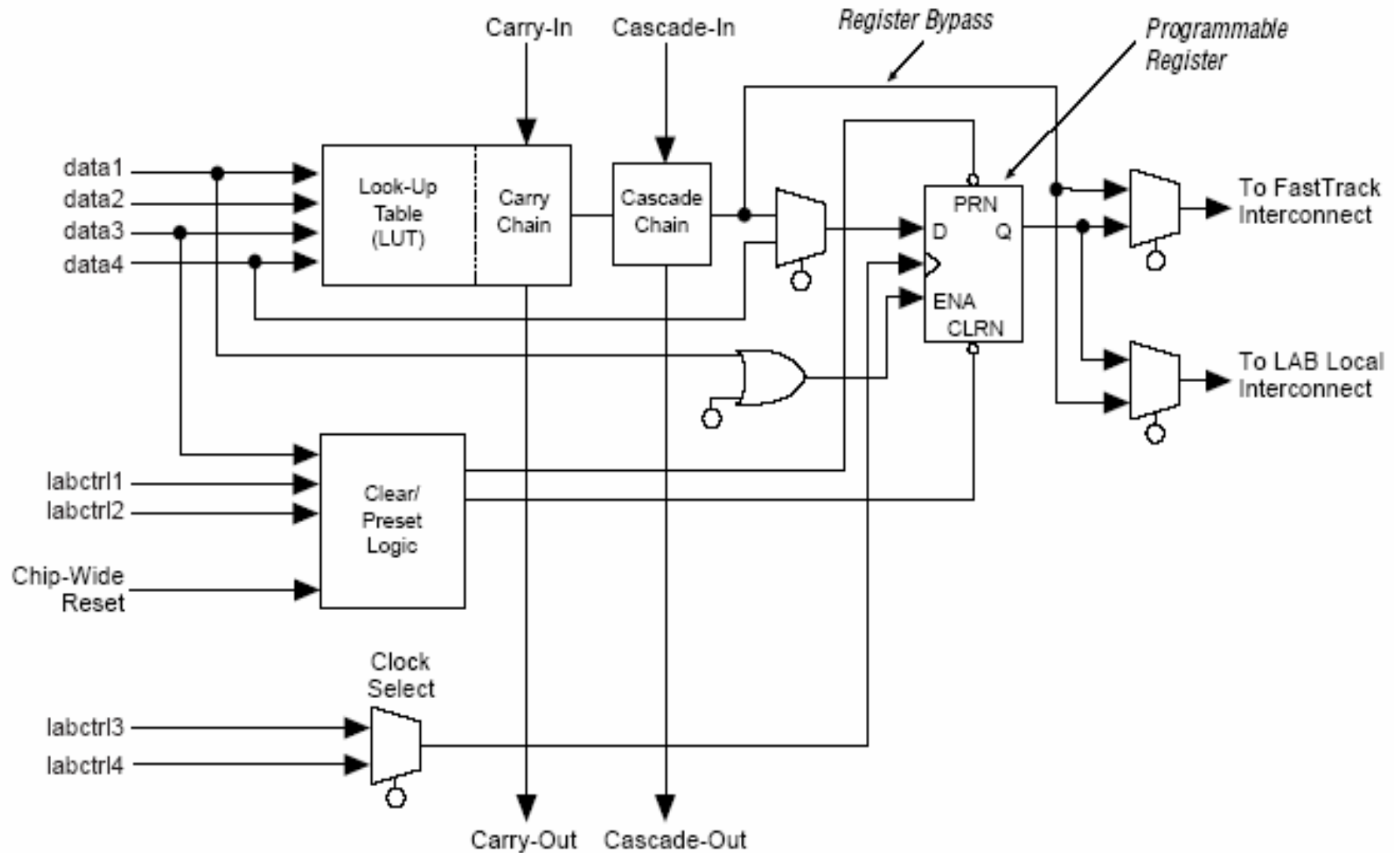
- Local Interconnect connects to local neighbors and to Column and Row interconnections

- Column, Row Interconnects used for long distance

- Channel intersections have switch matrix

- ❖ IOE- (I/O Element) connect to pins, usually have some additional FFs in block

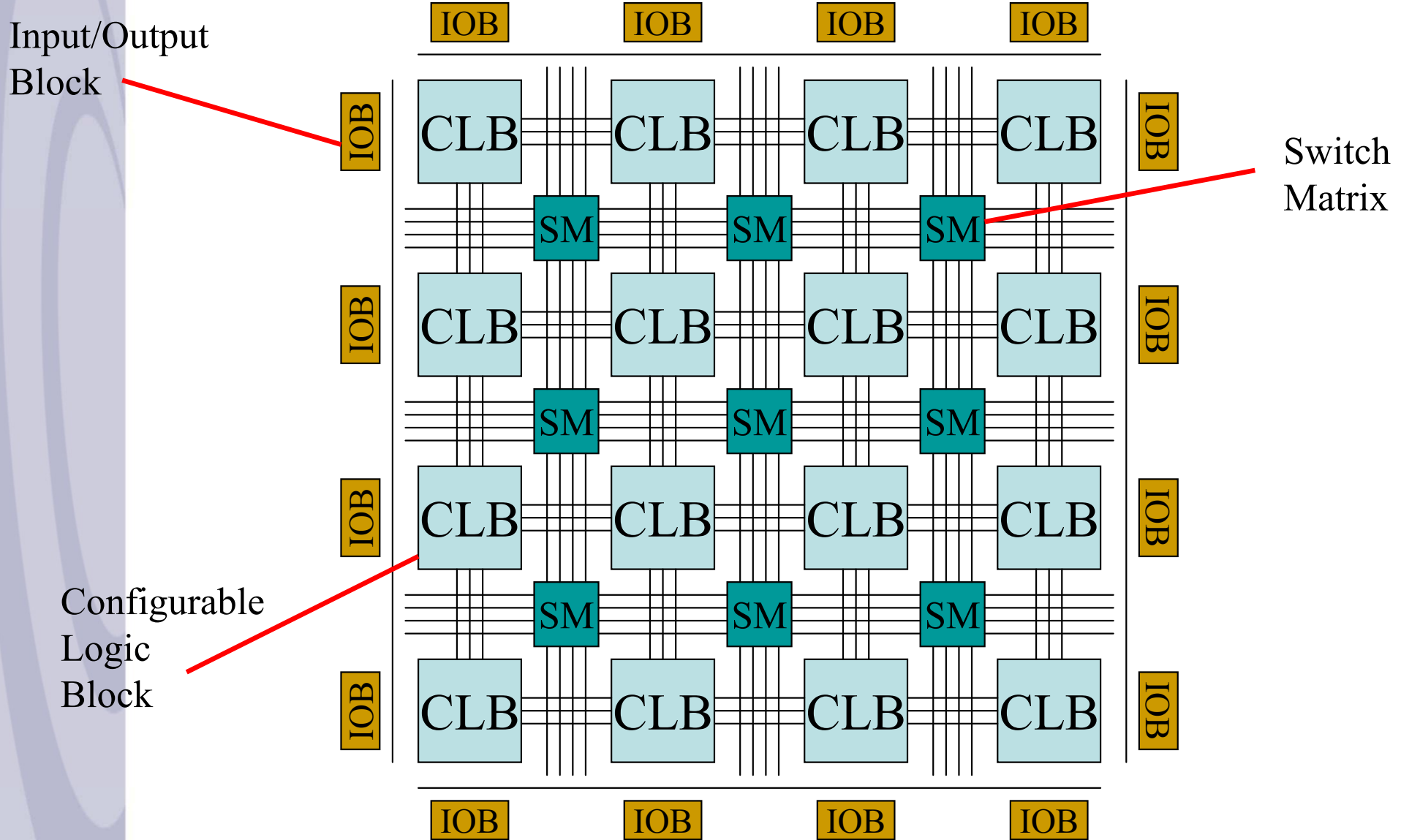
Altera Flex 10k Logic Element



Xilinx - FPGA Structure

- Typical organization in 2-D array
 - Configurable logic blocks (CLBs) contain functional logic
 - Combinational functions plus FFs
 - Complexity varies by device
 - CLB- interconnect is either local or long line
 - CLBs have connections to local neighbors
 - Horizontal and vertical channels use for long distance
 - Channel intersections have switch matrix
 - IOB- (I/O logic Blocks) connect to pins
 - Usually have some additional C.L./FF in block

Xilinx - FPGA Structure



Xilinx - FPGA IOB Structure

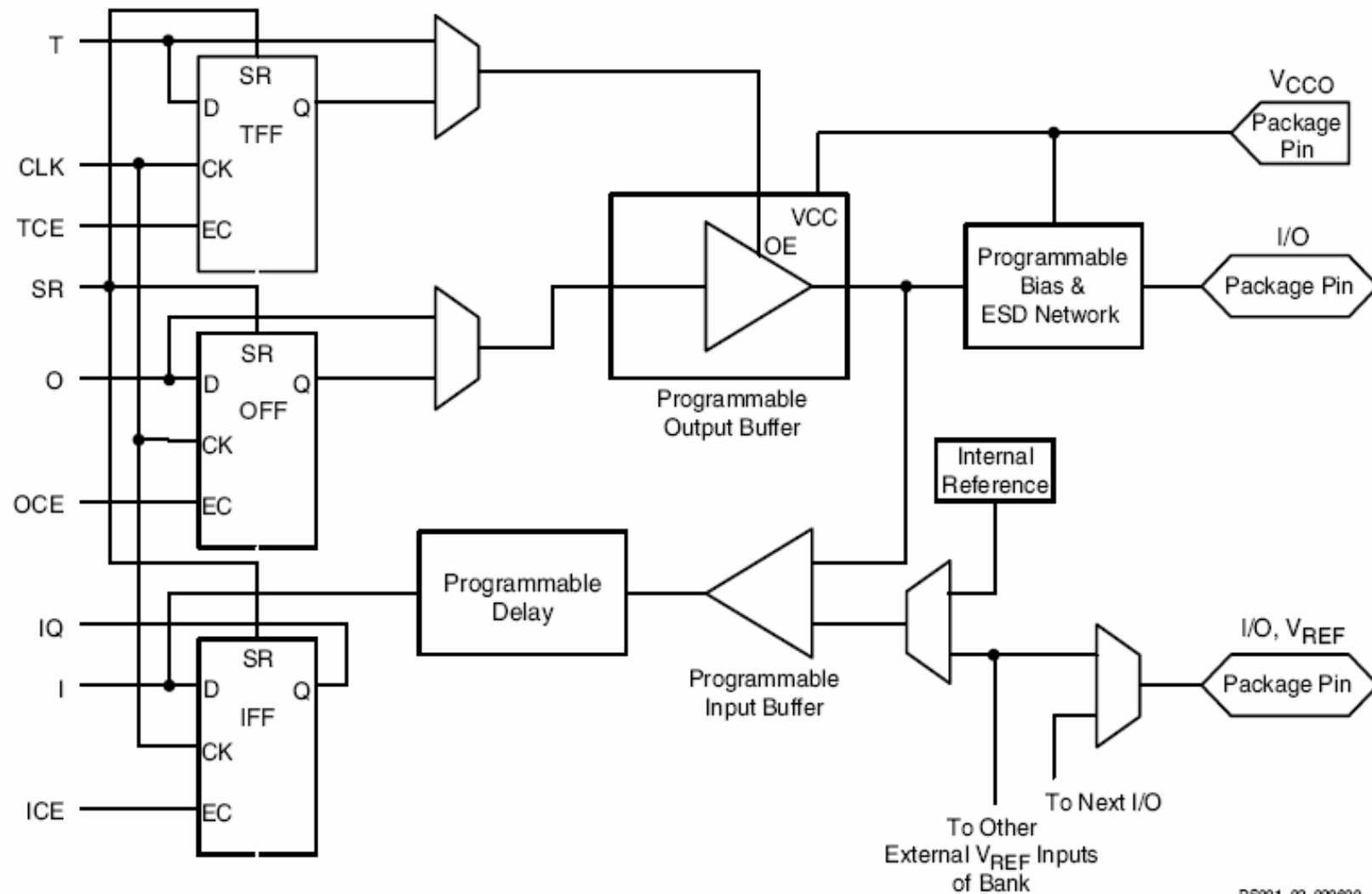
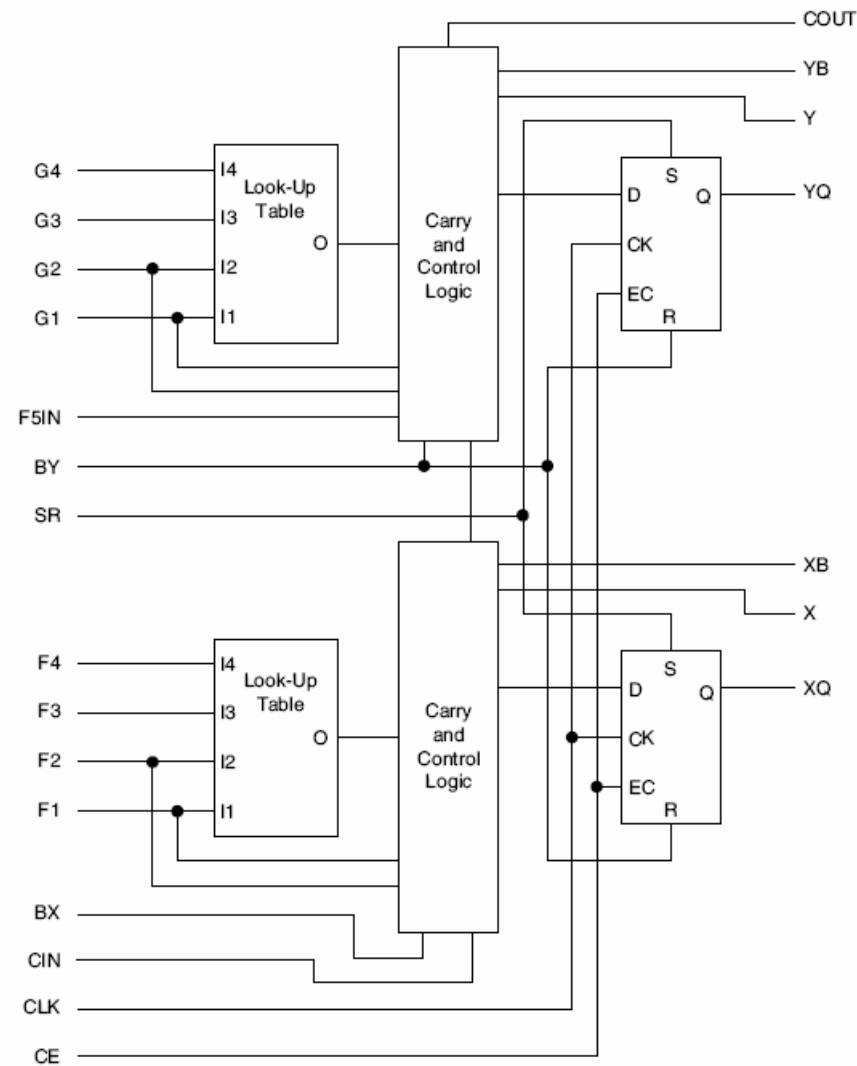


Figure 1: Spartan-II Input/Output Block (IOB)

Xilinx - FPGA CLB Structure



DS001_04_091400

Figure 3: Spartan-II CLB Slice (two identical slices in each CLB)