CSE 4205
Digital Logic Design

ROM, PLA, PAL, ALU

Course Teacher: Md. Hamjajul Ashmafee

AP, CSE, IUT

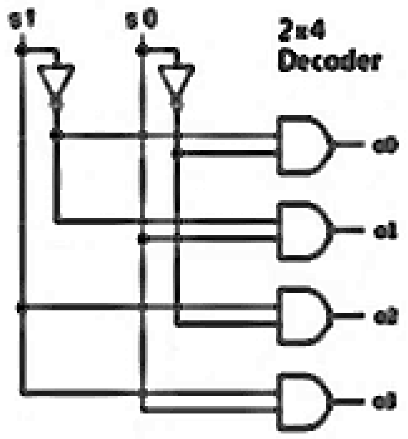
Email: ashmafee@iut-dhaka.edu

Introduction

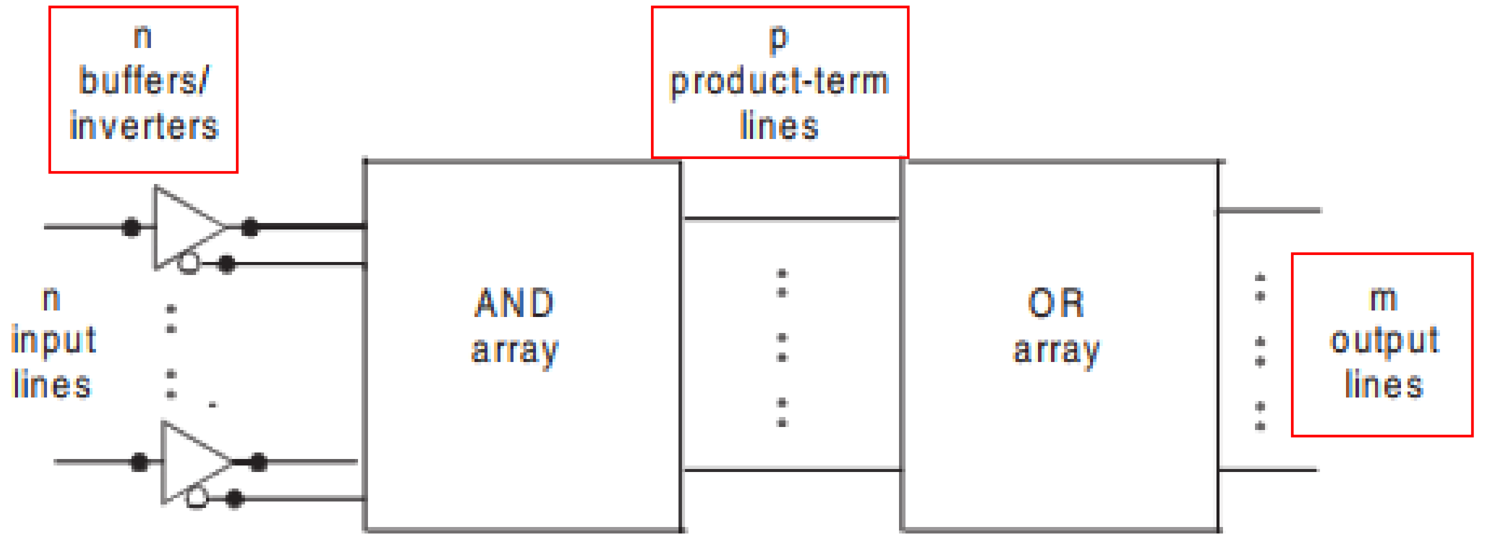
- With the advent of **large-scale integration** technology, it is feasible to fabricate large circuits within a single chip
  - A consequence – Programmable Logic Devices (PLD)
    - A kind of LSI
  - A combinational circuit to implement a set of Boolean functions

Concept to Produce PLDs

- Normally Boolean functions are expressed by **sum of product** forms or **sum of required minterms**
  - A **decoder** can be used to generate **2^n** minterms for **n** inputs
    - One or more OR gates are required to generate the expected function
- Specially they are used as a **memory device**
  - In spite of being **combinational circuit** devices



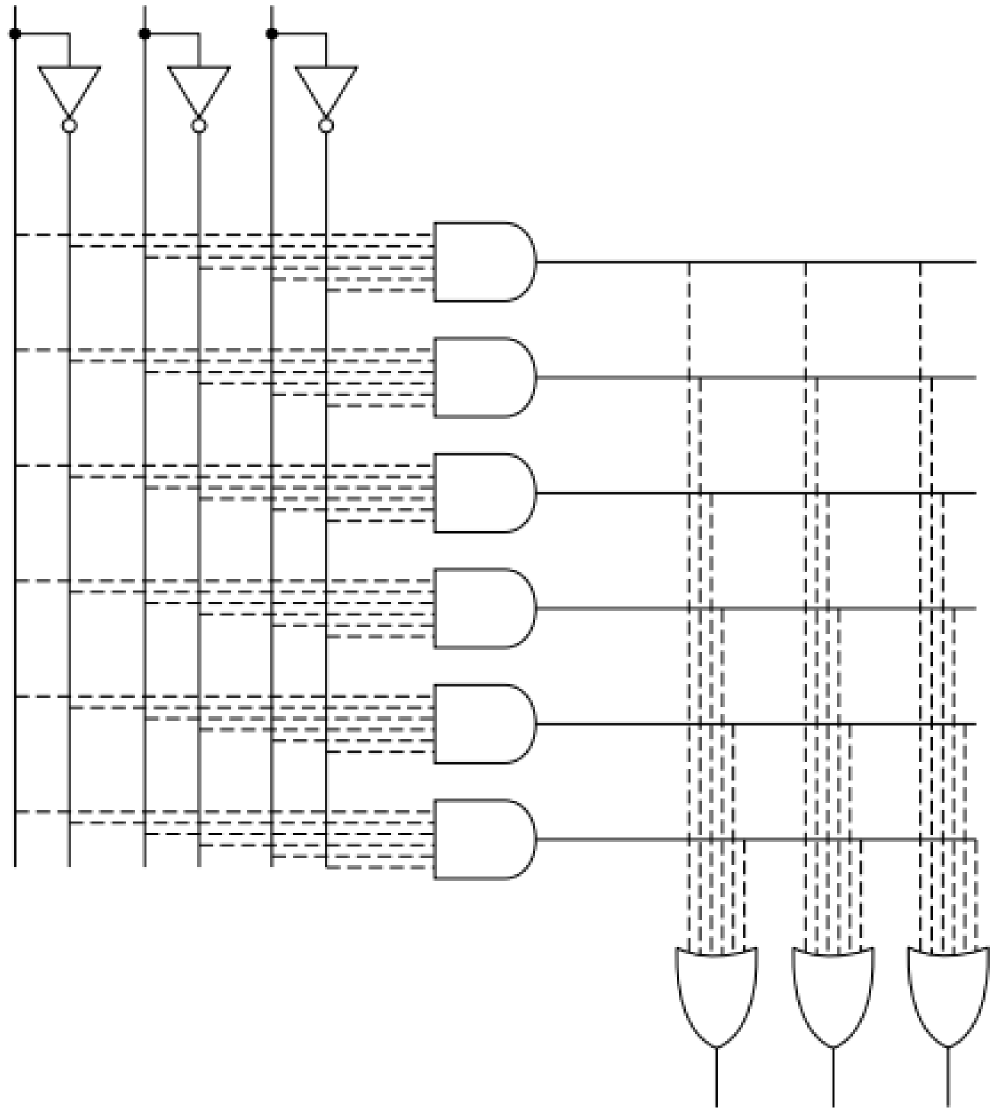
General Structure of PLD



Product term generation

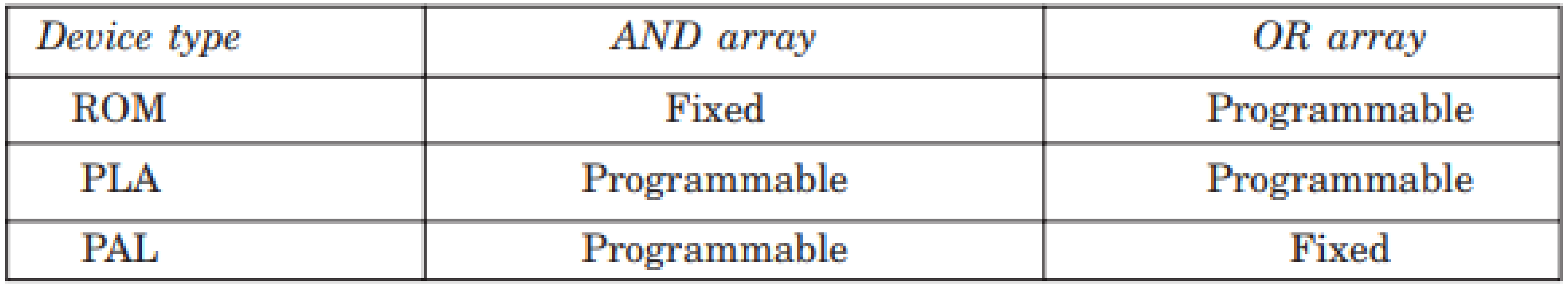
Sum term generation

Internal Structure of PLD



PLD Classification and Summery

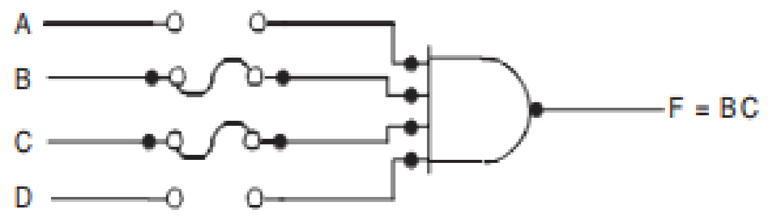
- Read Only Memory (ROM)
- Programmable Logic Array (PLA)
- Programmable Array Logic (PAL)



Device type	*AND array*	*OR array*
ROM	Fixed	Programmable
PLA	Programmable	Programmable
PAL	Programmable	Fixed

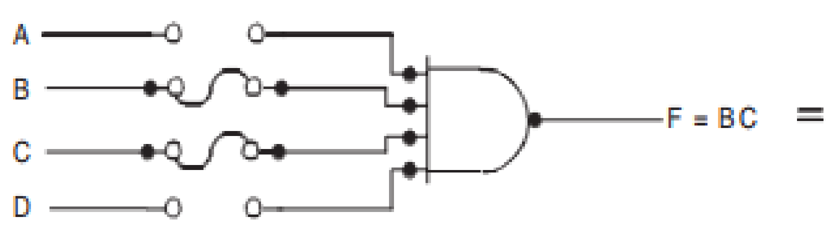
Concept: Programmable Array

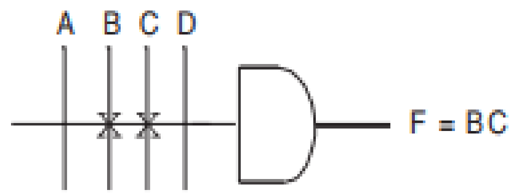
- In a **programmable array**, the **connections** of gates are **flexible**
  - To implement a **programmable gate**, a **fuse link** is employed
  - **Example:** a programmable input to an **AND gate**
    - Some of the fuses are blown out (removed/broken) to achieve the desired output of the gate

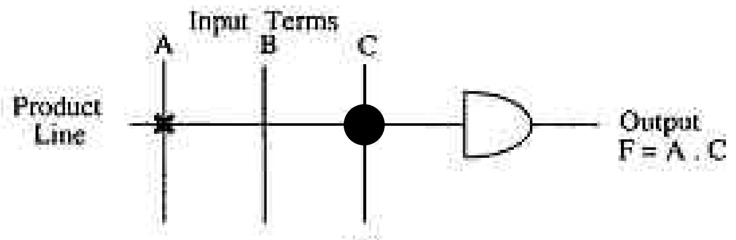


PLD Notation

- To simplify the **representation** of PLD connection:
  - A **single line** is drawn into the **logic gate** where the inputs are shown with **perpendicular lines**
  - The connected inputs are indicated by **cross (x)** at the junctions [fusible] and unconnected inputs are left **blanks**
  - Sometimes junctions are presented with **'bold' dots indicating permanent junctions**



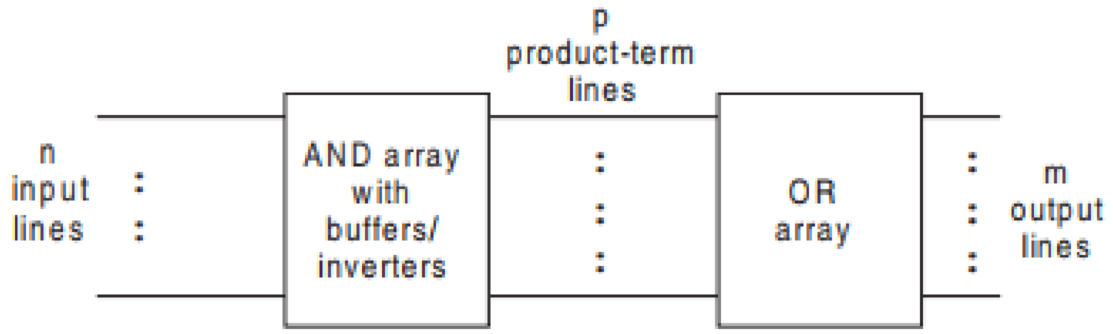




PLD Types in Details

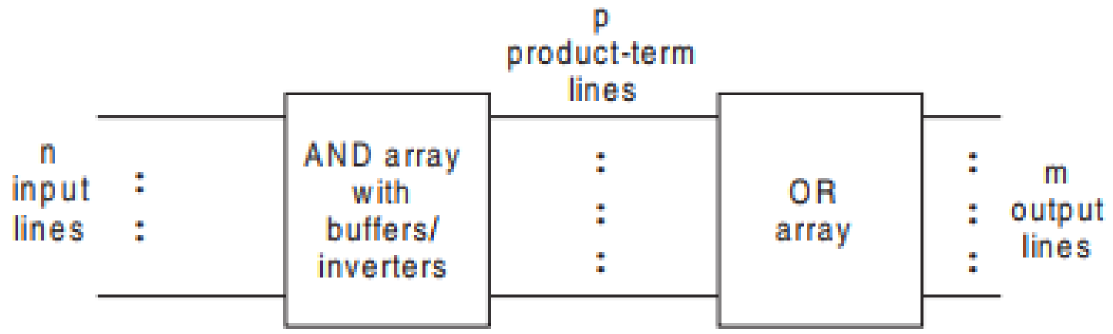
Read Only Memory (ROM)

- Common Notation – ***2^n x m*** ROM
  - n input lines and m output lines
    - 2^n intermediate ANDed (***i.e.*** product/minterm) terms which are **fixed**
    - m outlines lines are **programmable**
  - **Also, this notation is considered as total number of bits stored** as a **memory device.**
    - ***2^n*** denotes the **total number of addresses**
    - ***m*** denotes **the total number of bits per word**
  - A **two level logic representation** in **sum of minterms** form
  - **Block Diagram:**



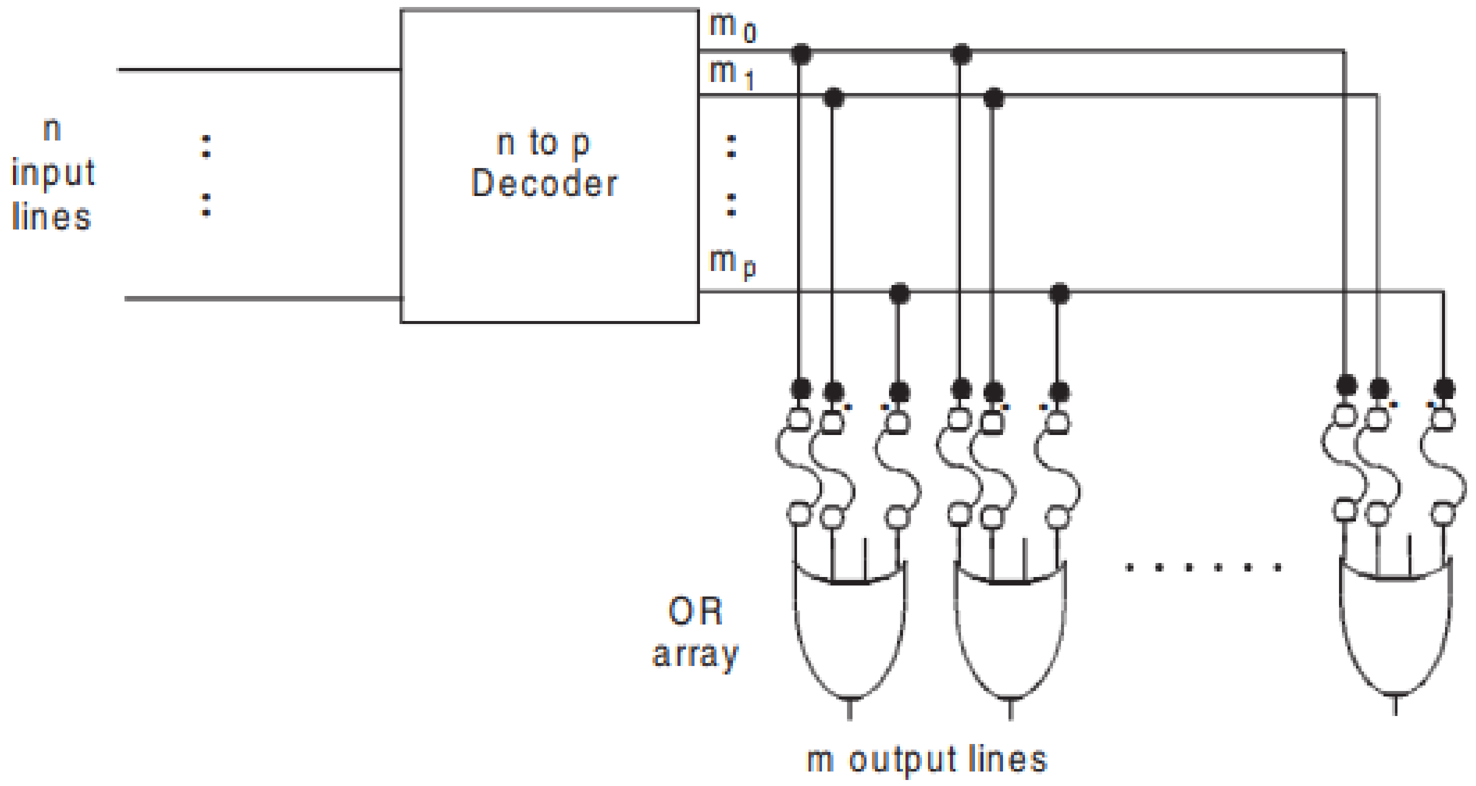
ROM with Decoder

- A **decoder** could be used to implement a **ROM**
    - Each minterm then is applied to required number of **OR gates** using **fusible** links which could be **blown up** selectively as per requirement (shown in logic diagram next)

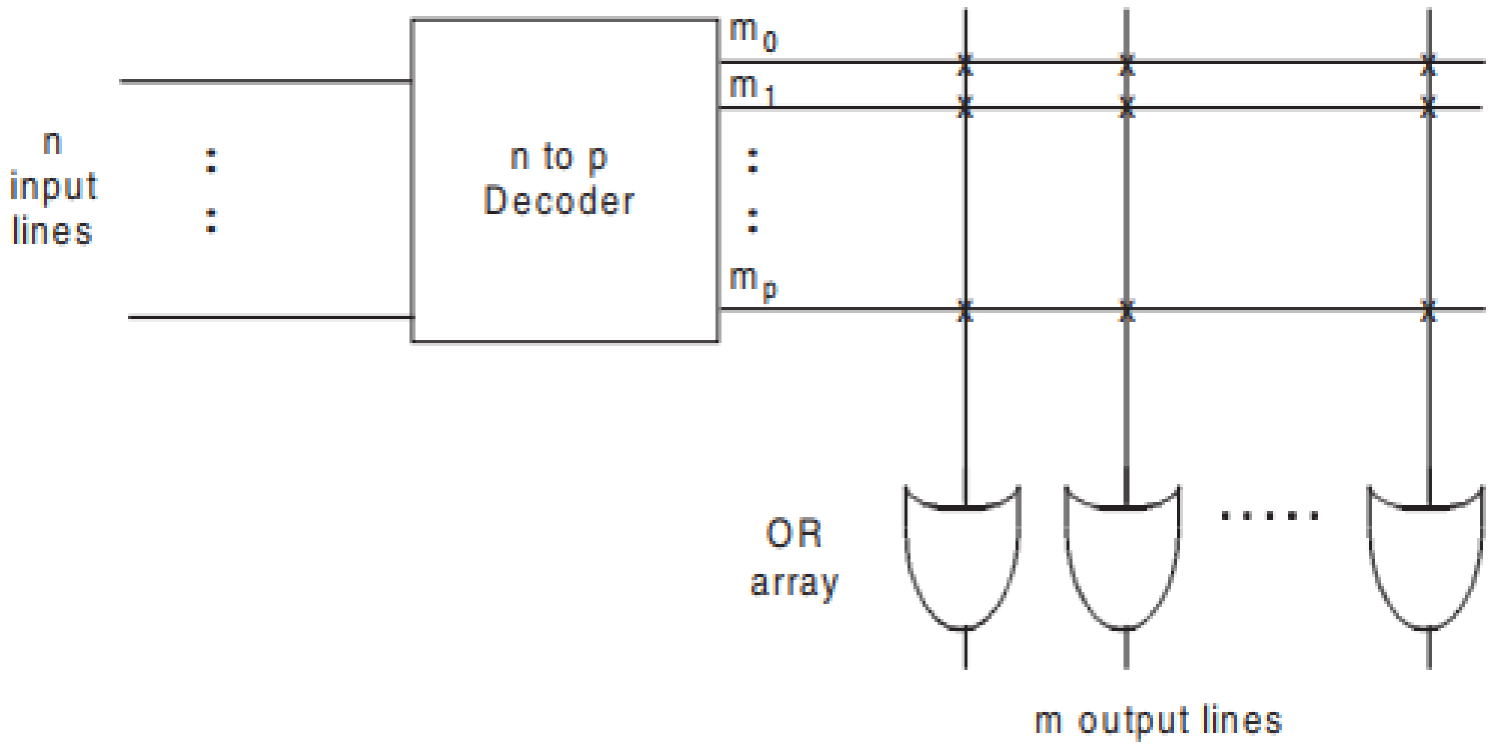


Decoder

2^nxm ROM: Logic Diagram



2^nxm ROM: PLD Notation



ROM: Example

Too simple to implement with a ROM

$F_1\ (A, B, C) = \ (\ 0,1,2,5,7)$

$F_2\ (A, B, C) = \ (1,4,6).$

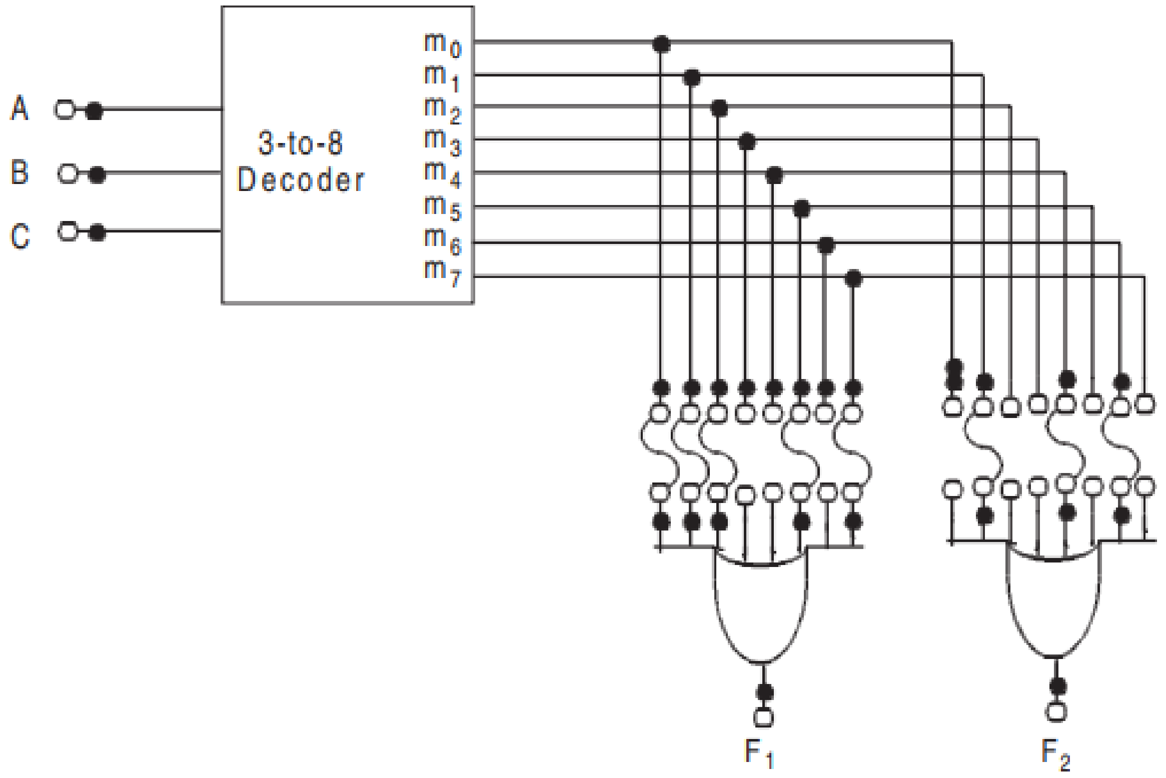
i.e. a **2³x2** or **8x2** ROM is required.

Program table or truth table for ROM:

Decimal	*Input Variables*			*Outputs*	
Equivalent	*A*	*B*	*C*	F_1	F_2
0	0	0	0	1	0
1	0	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	0	1
7	1	1	1	1	0

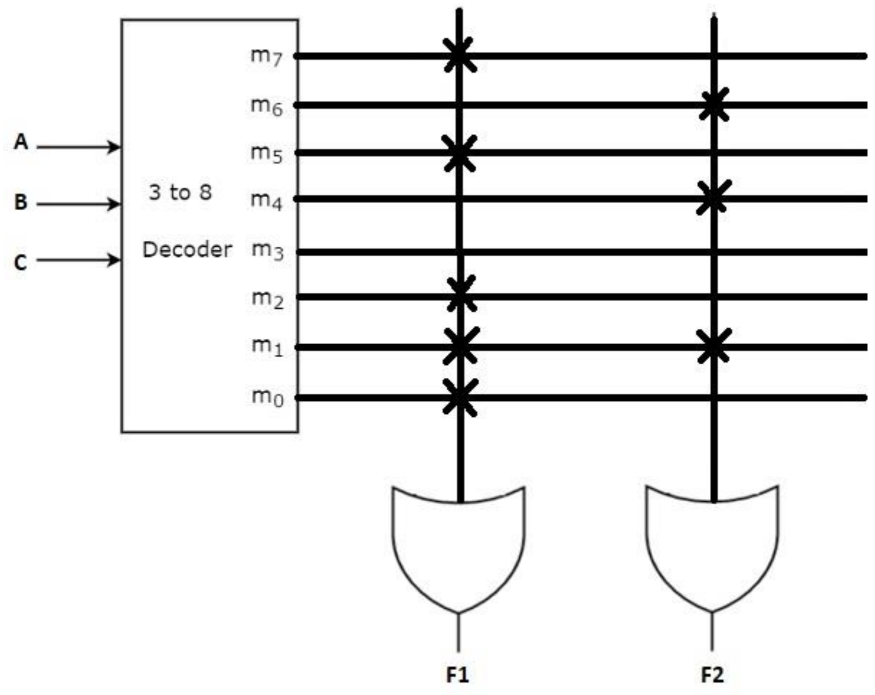
ROM: Example…

- **Logic diagram:**



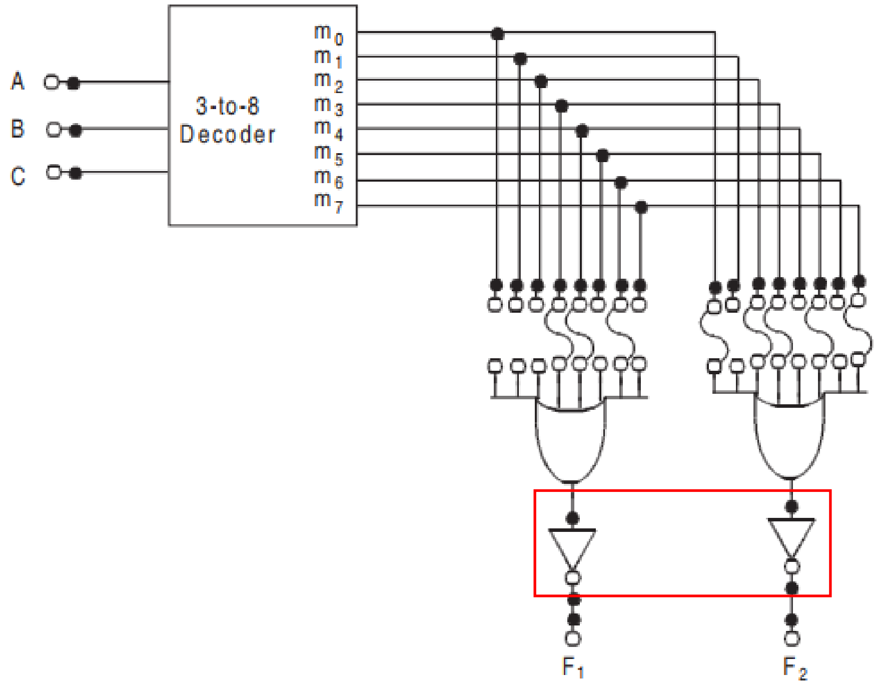
ROM: Example…

- **Logic diagram with PLD notation :**



ROM: Example...

- Some ROMs are available with **inverter logic gate (NOT)** with each OR gates



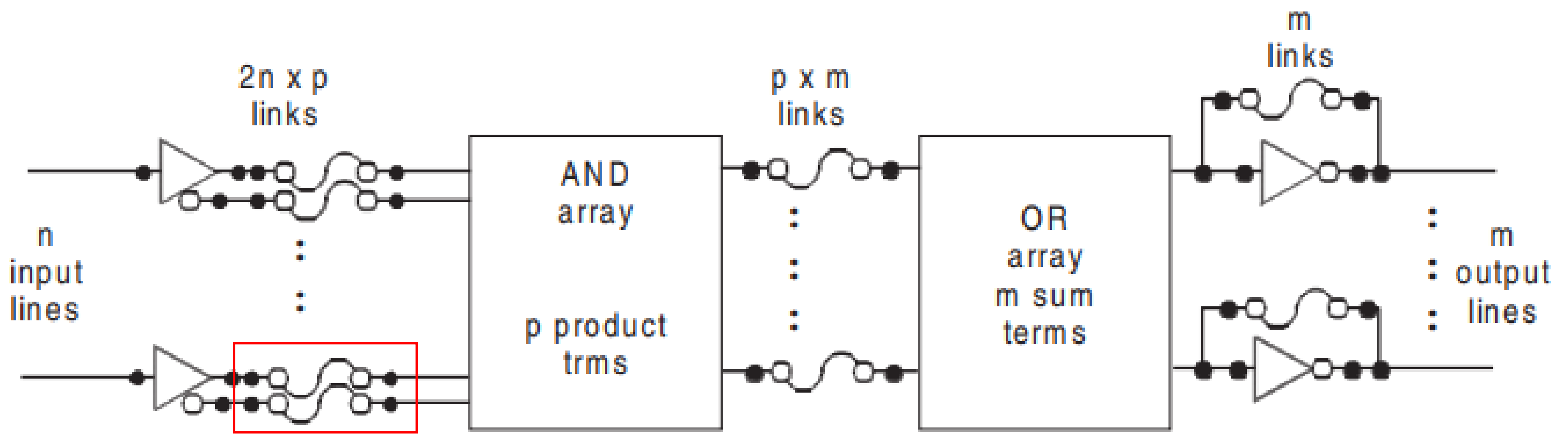
Programmable Logic Array (PLA)

- **Motivation:**
  - A combinational circuit may contain ***don't care conditions*** which will never occur
    - A **waste of memory** considering ***don't cares*** as valid ***minterms***
    - **Example:** Code conversion from BCD to Excess 3 where **6 addresses will not be used** for a 16x4 ROM
      - A waste of ***6x4*** bit locations
  - PLA (kind of LSI) is **more economical** to use where don't care cases are **excessive**

Concept of PLA

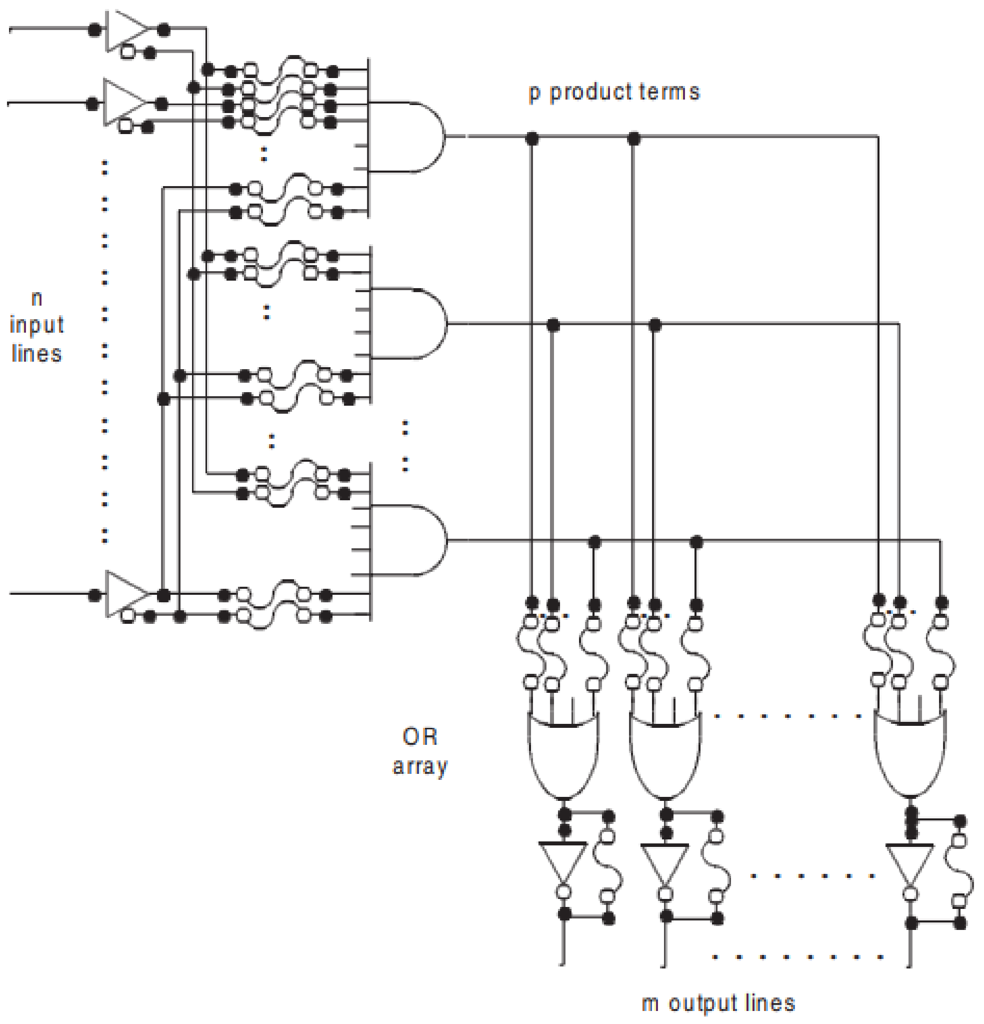
- Common notation: **n** x **p** x **m**
  - **n** = number of inputs
  - **p** = number of product terms
  - **m** = number of outputs (or sum terms)
- Total number of ***programmable links*** = (2n x p) +(p x m) + m
- The basic difference between ROM and PLA is in the **number of the product term**
  - ***i.e.*** 2^n (ROM) >> p (PLA)
  - ***e.g.*** 16 x **48** x 8 PLA. Here the number of product term is drastically reduced

PLA: Block Diagram



Programmable links

PLA: General Structure or Logic Diagram



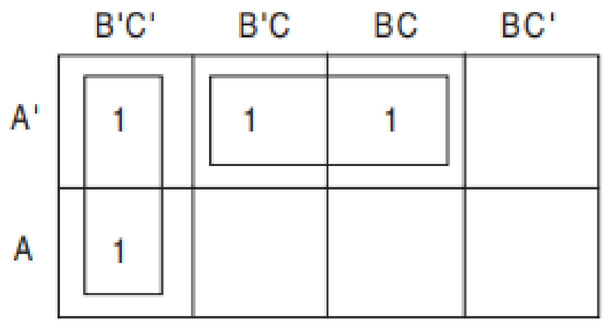
PLA: Example

Too simple to implement with a PLA

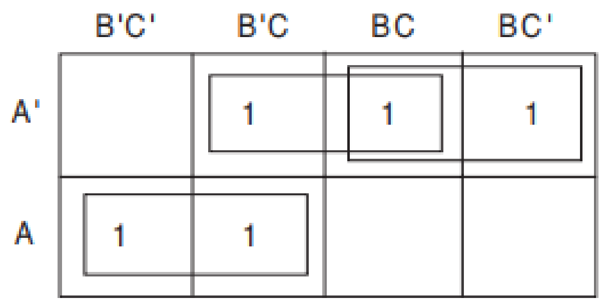
- **Example:** Use a *3 x 4 x 2* PLA to implement the following Boolean expressions

$$F_1\,(A, B, C) = (0, 1, 3, 4)$$
$$F_2\,(A, B, C) = (1, 2, 3, 4, 5).$$

- To get required and reduced product terms, **simplified expressions** are:



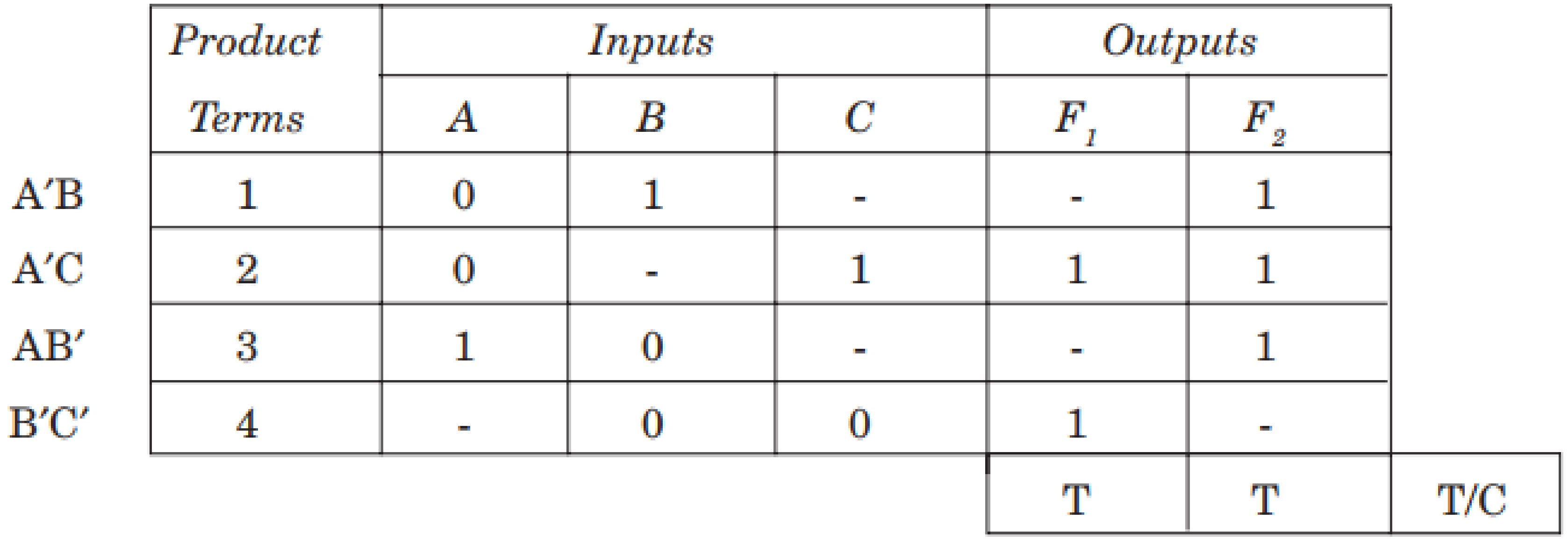
$$F_1 = B'C' + A'C$$



$$F_2 = A'B + A'C + AB'.$$

PLA: Example…

- **PLA program table**

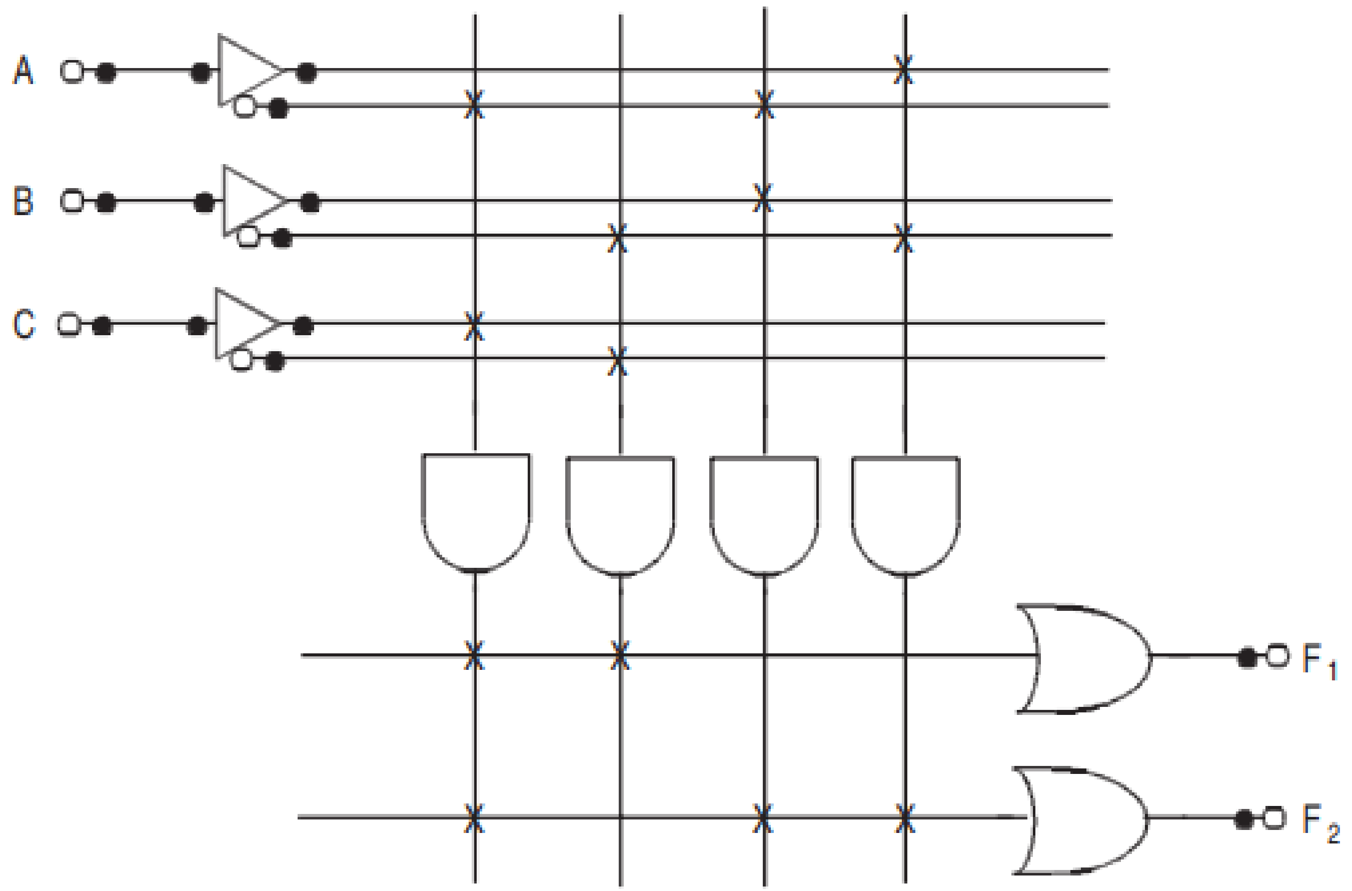


	Product Terms	*Inputs*			*Outputs*		
		A	*B*	*C*	F_1	F_2	
A′B	1	0	1	-	-	1	
A′C	2	0	-	1	1	1	
AB′	3	1	0	-	-	1	
B′C′	4	-	0	0	1	-	
					T	T	T/C

T = in its original form, inverter is blown up
C = in its complement form, inverter is used

PLA: Example…

- **Internal connections of PLA** with PLD notation:



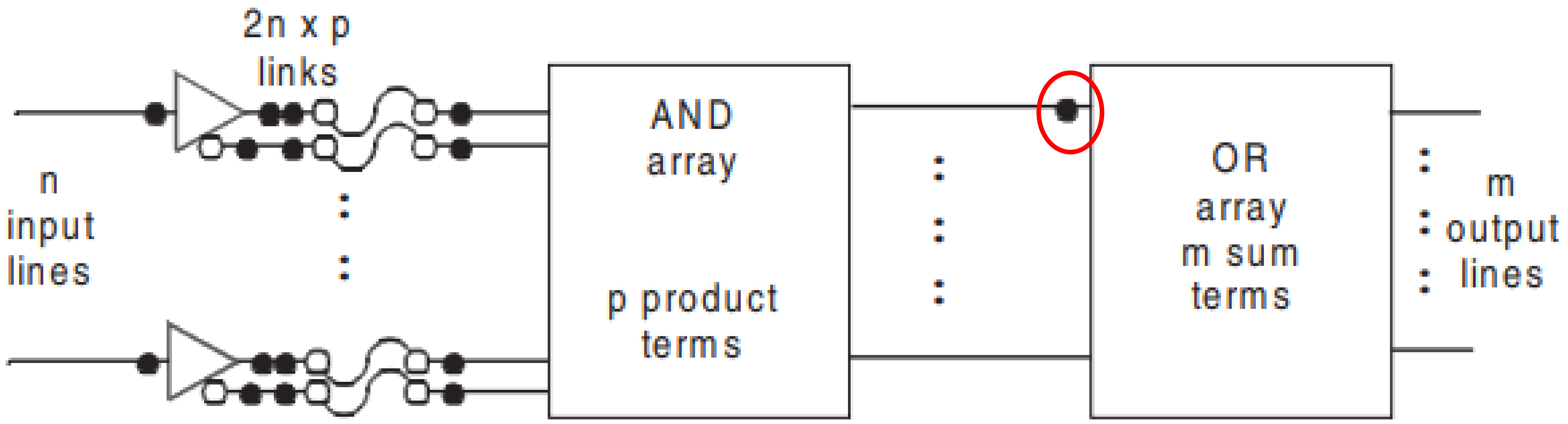
Programmable Array Logic (PAL)

- General structure of PAL is very similar to PLA
  - **But,** only difference between these is in a PAL device, only **AND gates are programmable** whereas **OR gates are fixed** by manufacturer.
    - So, PLA is less expensive, **more flexible** and more easier than PAL

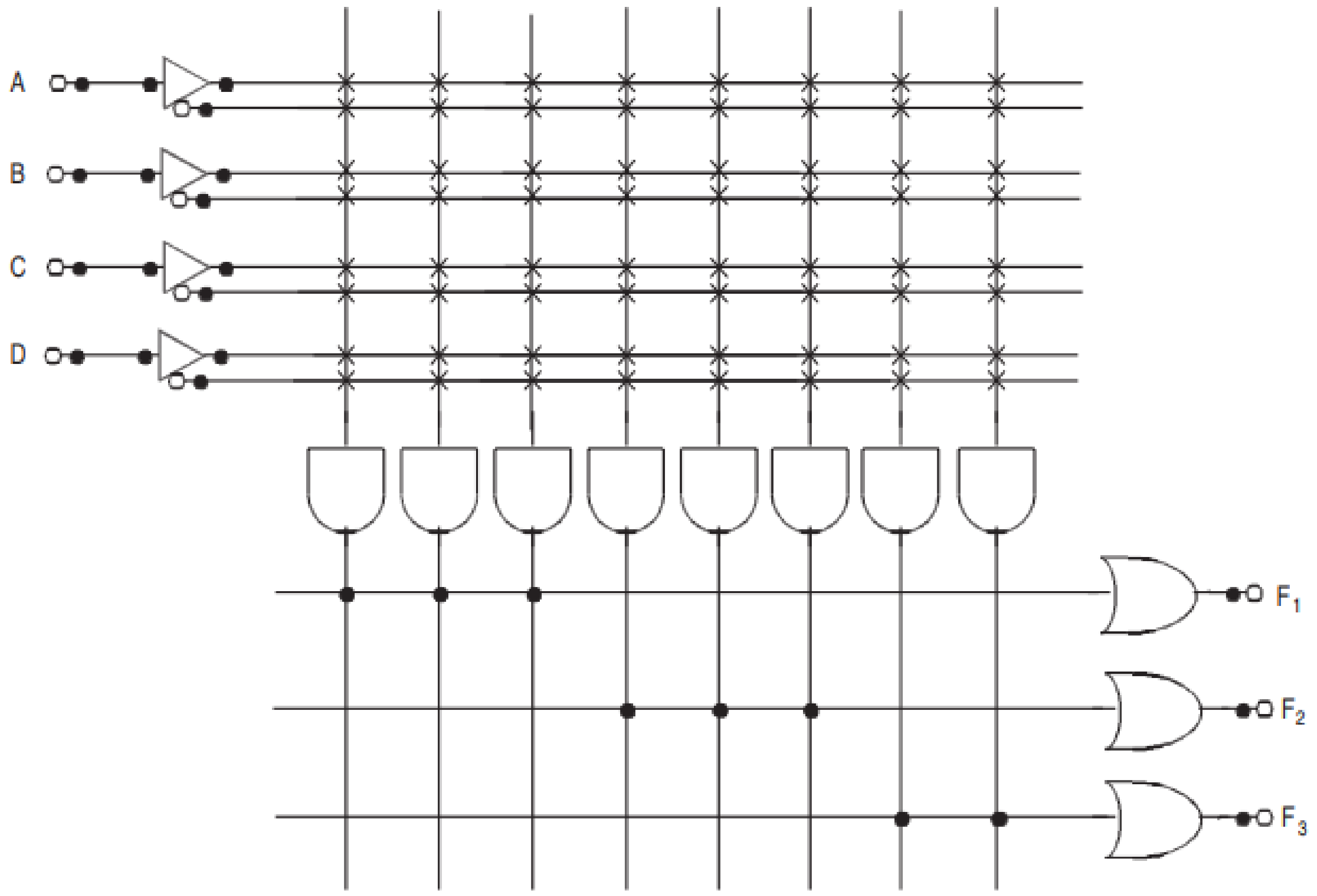
Concept of PAL

- External inputs are connected with the AND gates with programmable links
  - Whereas, **the output of AND gates are fed to OR array with fixed connections**
- ***All the AND outputs are not fed to all OR gates***
  - Rather, few AND outputs are connected to an OR gate based on the manufacturer's direction (Figure shown)
- Similar **n** x **p** x **m** notation for PAL as PLA:
  - **n** = number of inputs
  - **p** = number of product terms
  - **m** = number of outputs (or sum terms)

PAL: General Structure



A Generic 4 x 8 x 3 PAL



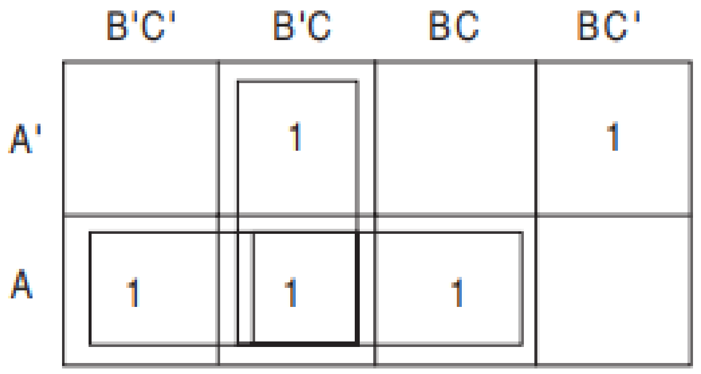
PAL: Example with a 4 x 8 x 3 PAL

Too simple to implement with a PAL

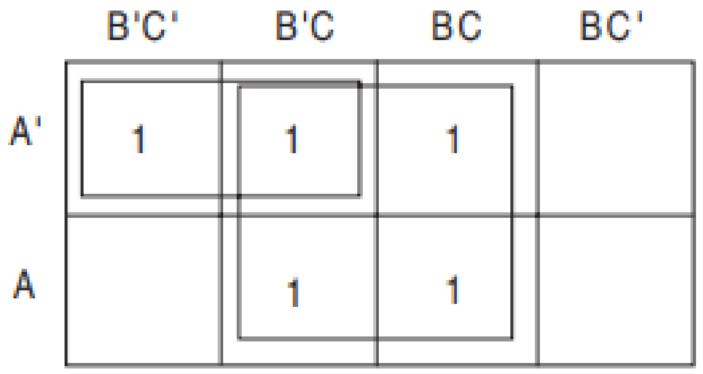
- **Example:** Use a ***4 x 8 x 3*** PAL to implement the following Boolean expressions

$$F_1(A,B,C) = (1,2,4,5,7)$$
$$F_2(A,B,C) = (0,1,3,5,7)$$

- To get required and reduced product terms, **simplified expressions** are:



$$F_1(A,B,C) = AB' + AC + B'C + A'BC'$$



$$F_2(A,B,C) = C + A'B'.$$

PAL: Example with a 4 x 8 x 3 PAL…

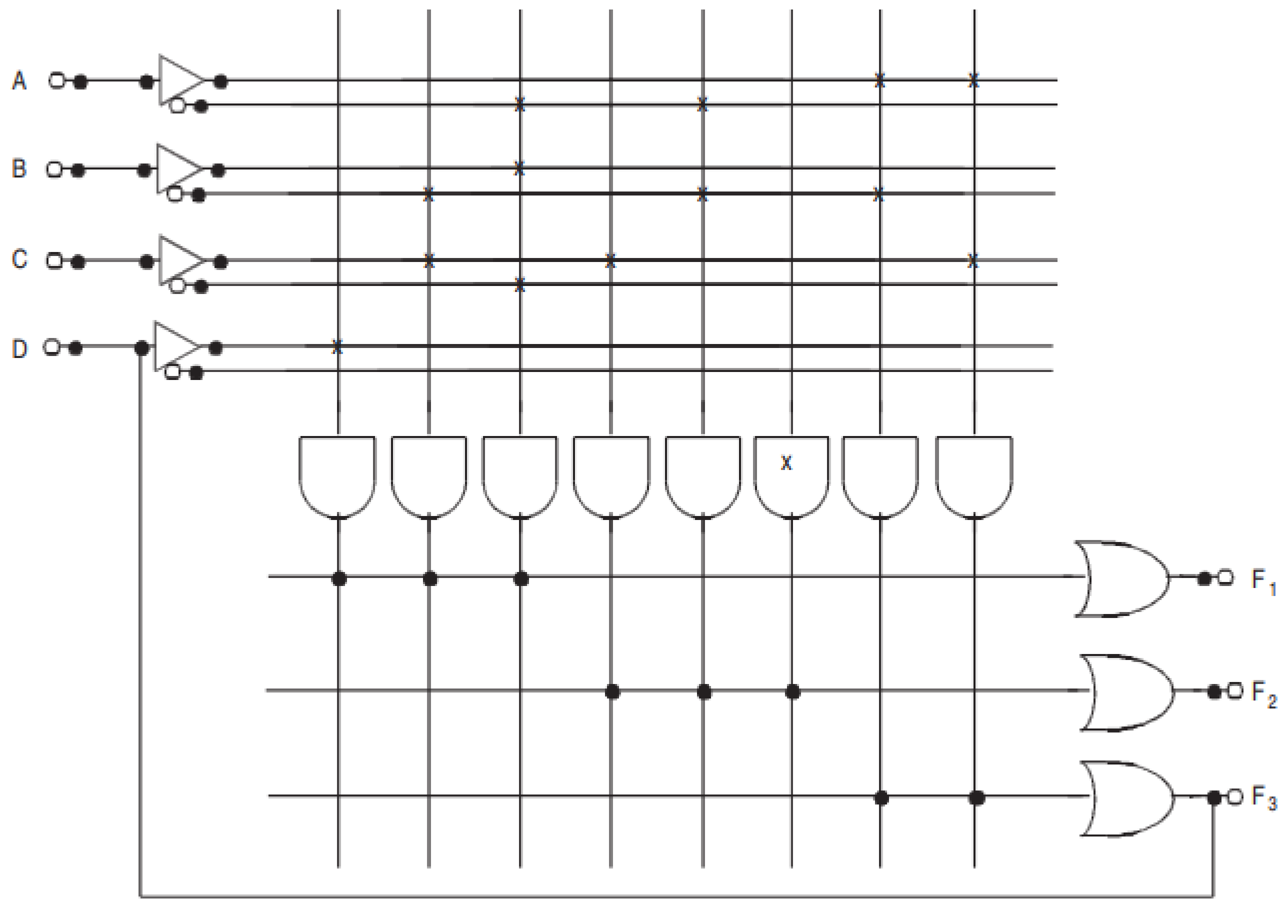
- But, in the specified PAL, there is **no** OR gate **with 4 product terms**
    - So alternative realization could be:

$$F_1(A,B,C) = F_3 + B'C + A'BC'$$
$$F_2(A,B,C) = C + A'B'$$

Where $F_3 = AB' + AC.$

- Now these three functions could realize the specified Boolean expressions where F_3 (**sub-function**) has to be inserted as an input to the F_1 again

PAL: Example with a 4 x 8 x 3 PAL…

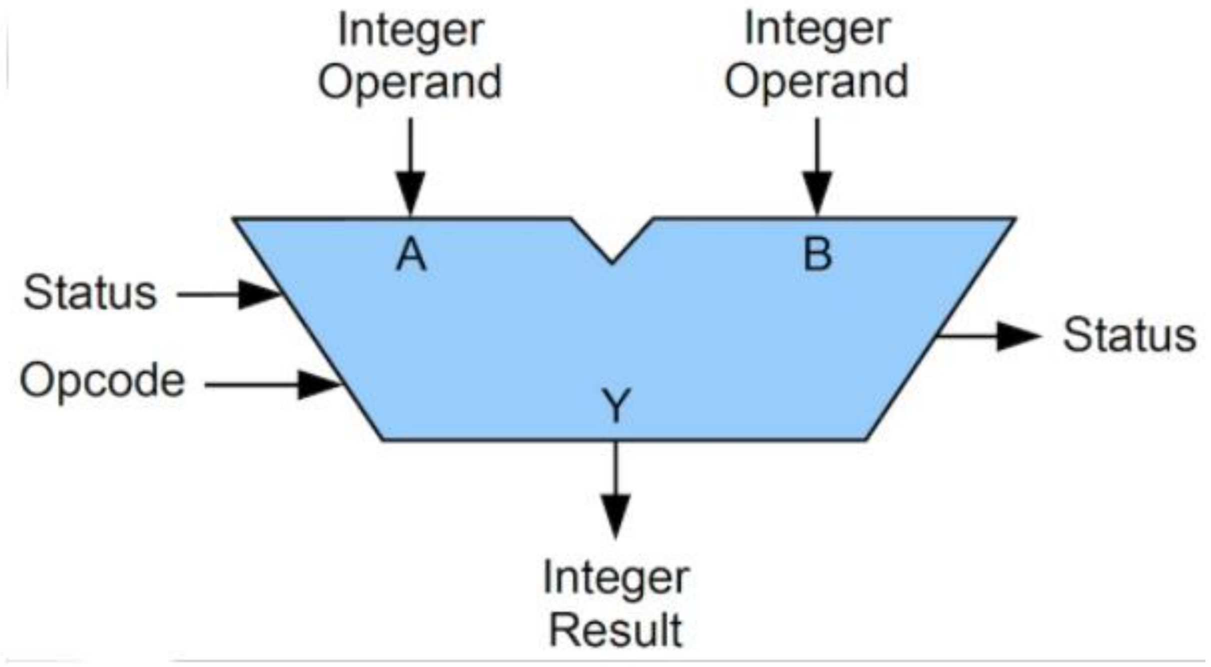


Other PLDs

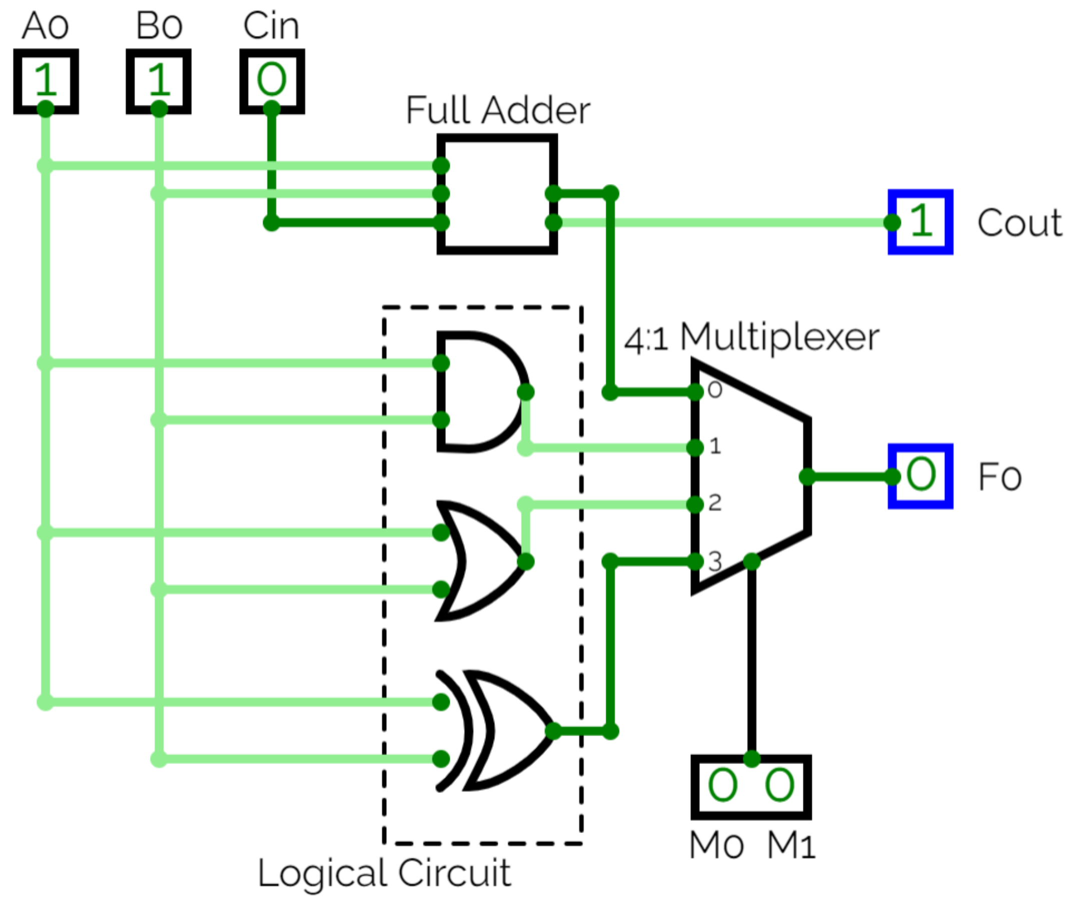
- Registered PAL devices
  - For **sequential circuits**
- **Generic Array Logic** Devices (GAL)
- **Field Programmable Gate Array** (FPGA)

ALU as Logic Device

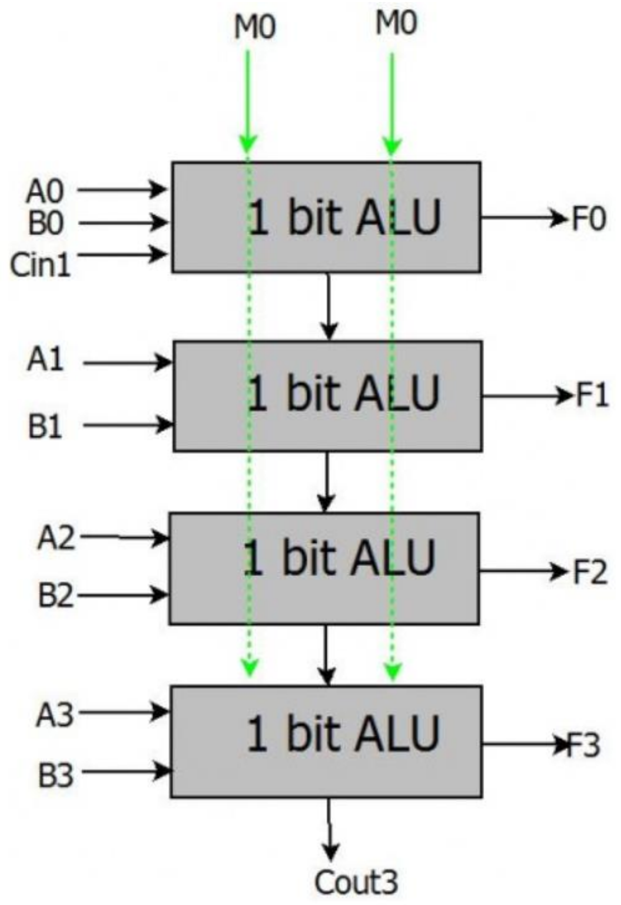
- ALU is a part of CPU that performs the **calculation** and **condition testing**
  - Considered as fundamental **building block** of CPU



ALU: Example of ALU

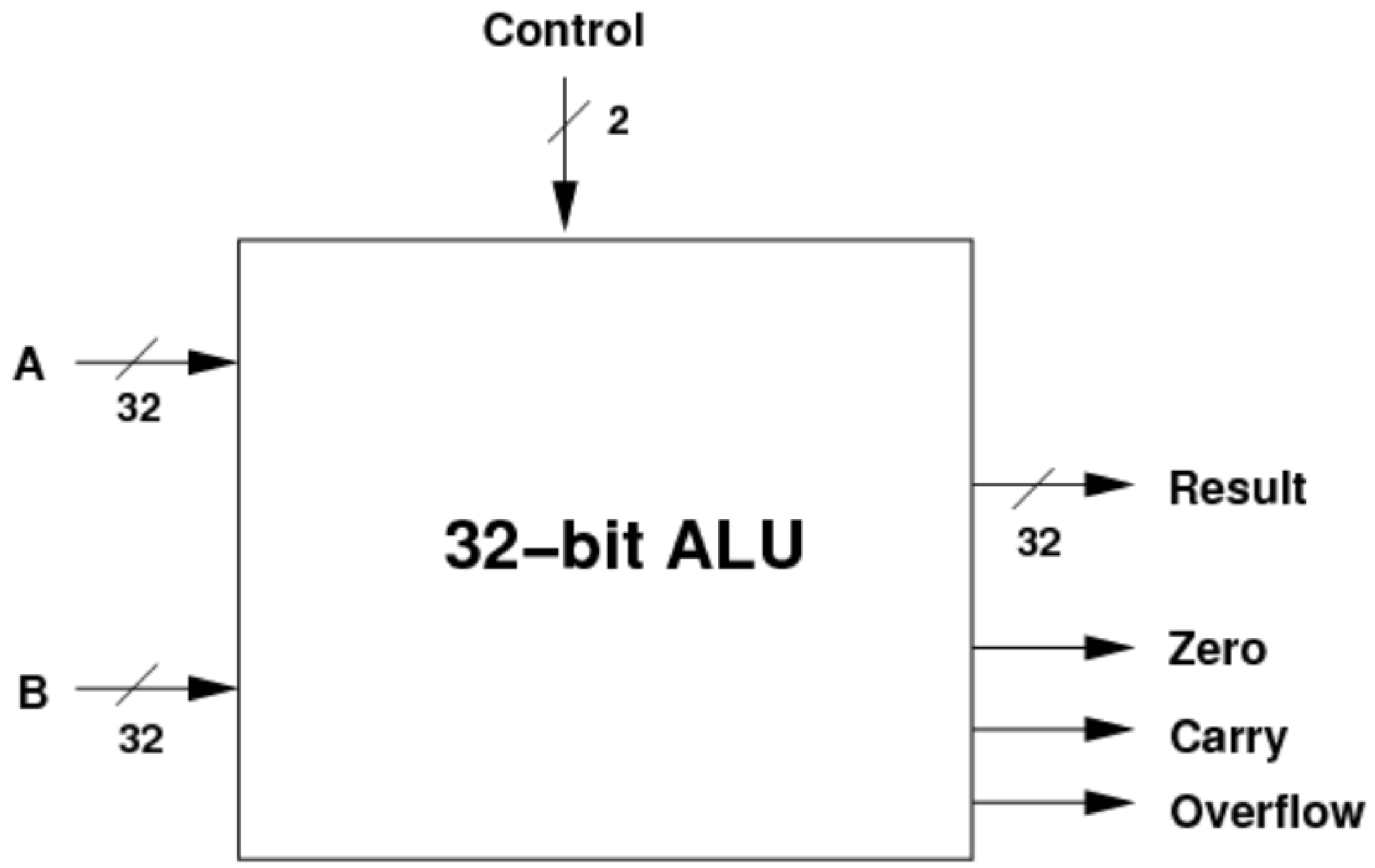


Circuit Diagram of 1-bit ALU



Block Diagram of multiple 1-bit ALUs

ALU: More Complex Example


Control
2
A
32
B
32
32–bit ALU
Result
32
Zero
Carry
Overflow