Lect 15

Programmable Logic

CS221: Digital Design

Dr. A. Sahu
Dept of Comp. Sc. & Engg.
Indian Institute of Technology Guwahati

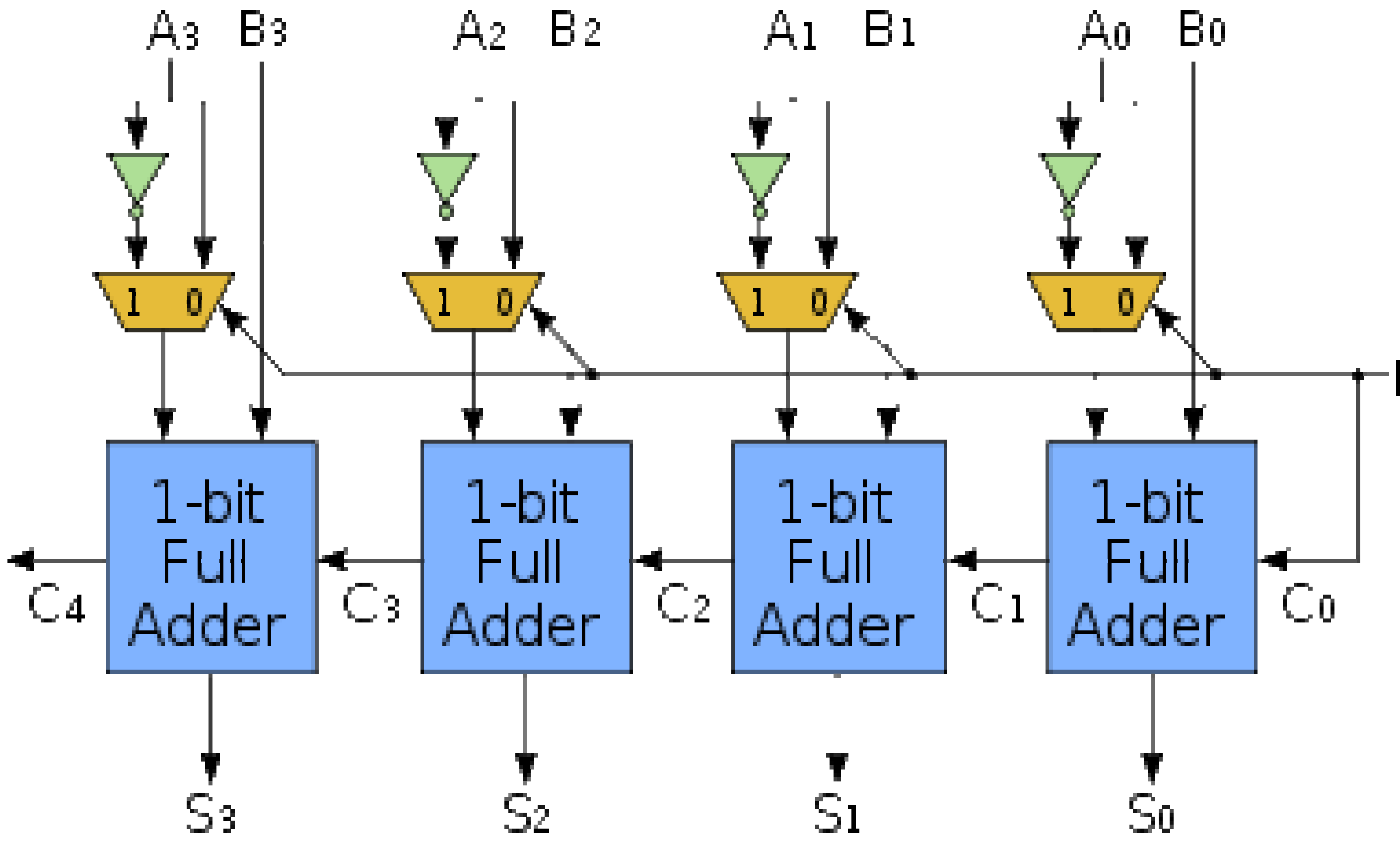
Outline

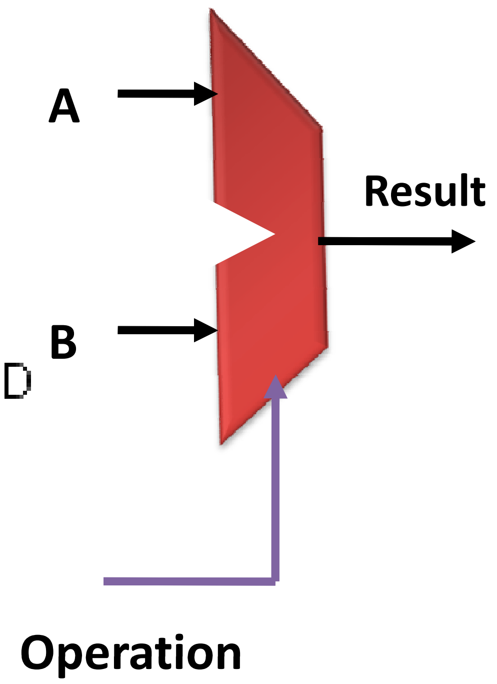
- Programmable Logic
- PAL, PLA,
- Memory
  - ROM, PROM, EPROM, EEPROM
  - SRAM : Memory Cell
- CPLD, CLB, FPGA
- FPGA/ASIC Design Flow
- HDL Programming : Verilog HDL

Programmable Logic Devices

Programmable Via Control : Adder/Substractor

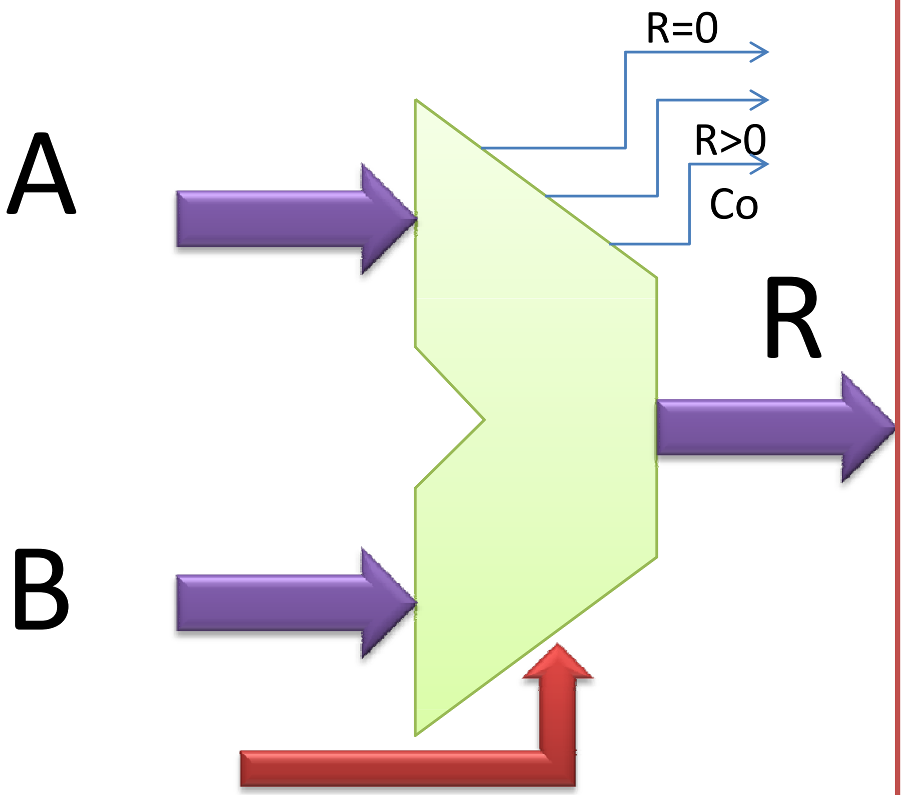
- C= B-A=B+(-A)=B+ (A^b+1), A^b is complement of A
- D is control bit: D=0/1 operation is add/sub





Programmable Via Select: ALU

- Arithmetic and Logic Unit
- Add/Sub/OR/AND/Shift…



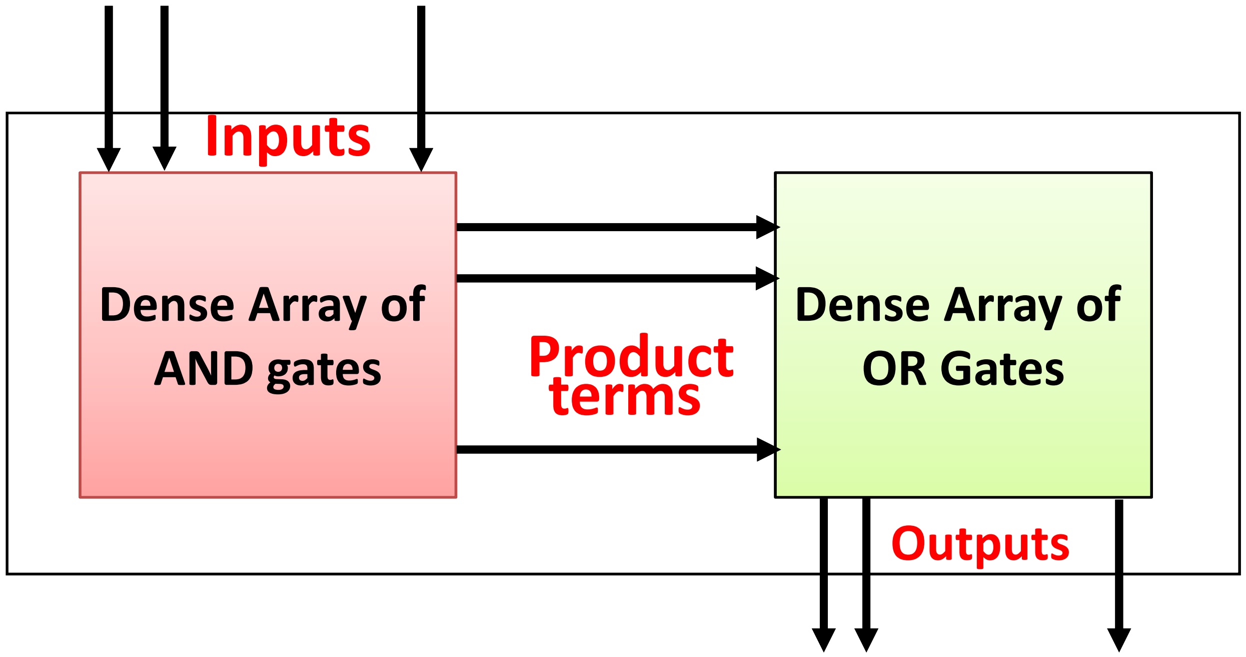
Control: What to do ?

if	Control=0	R =A +B
if	Control=1	R =A - B
if	Control=2	R =NOT A
if	Control=3	R =A AND B
if	Control=4	R =A OR B
if	Control=5	R =A XOR B
if	Control=6	R = (A<B)?0:A
if	Control=7	R =A SHFT B

Programmable Logic Devices

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

Example: *Equations*

F0 = A + B' C'
F1 = A 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
B' C	-	0	1	0	0	0	1
A C'	1	-	0	0	1	0	0
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

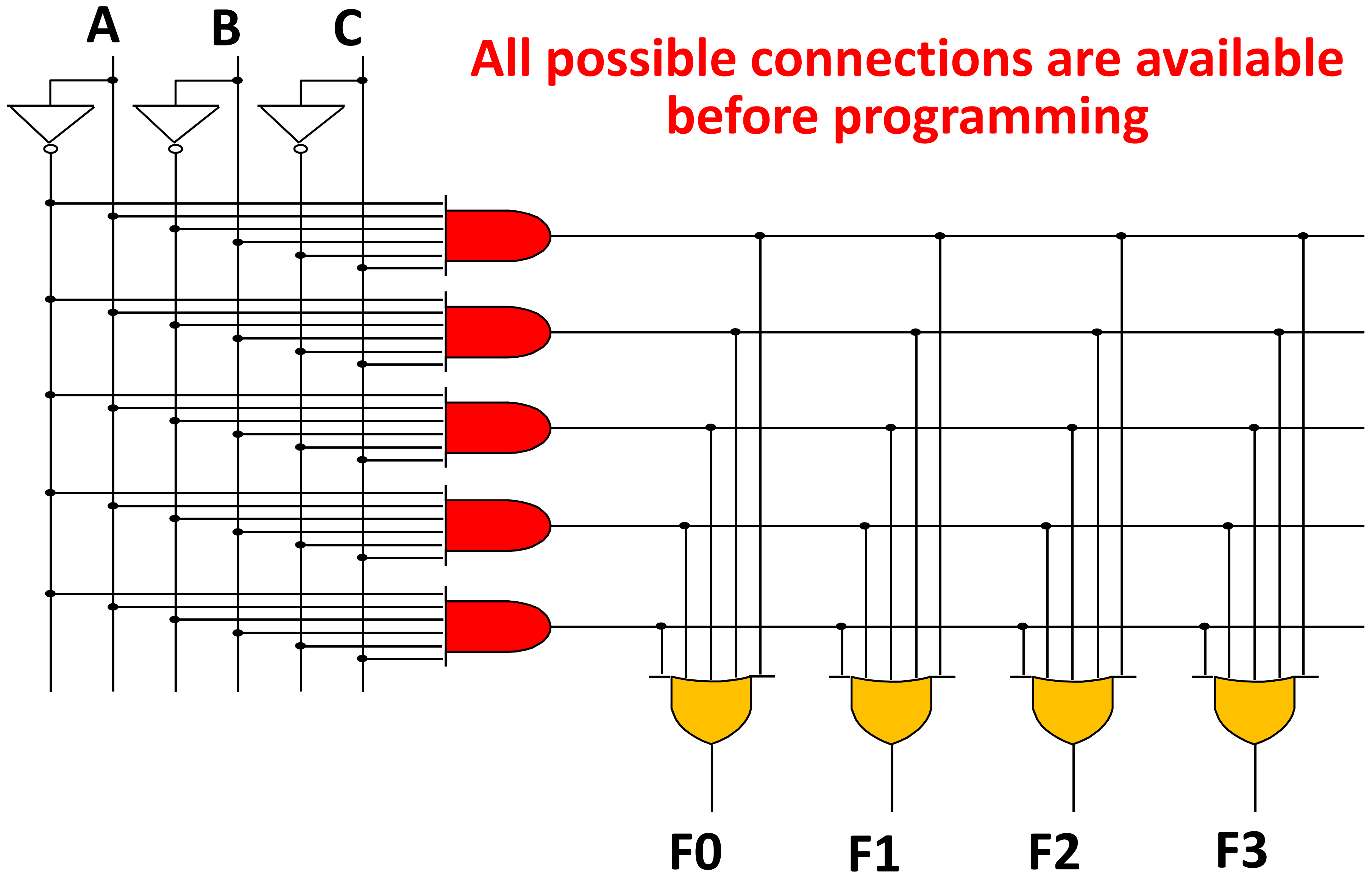
Output Side:

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

PLA Logic Implementation

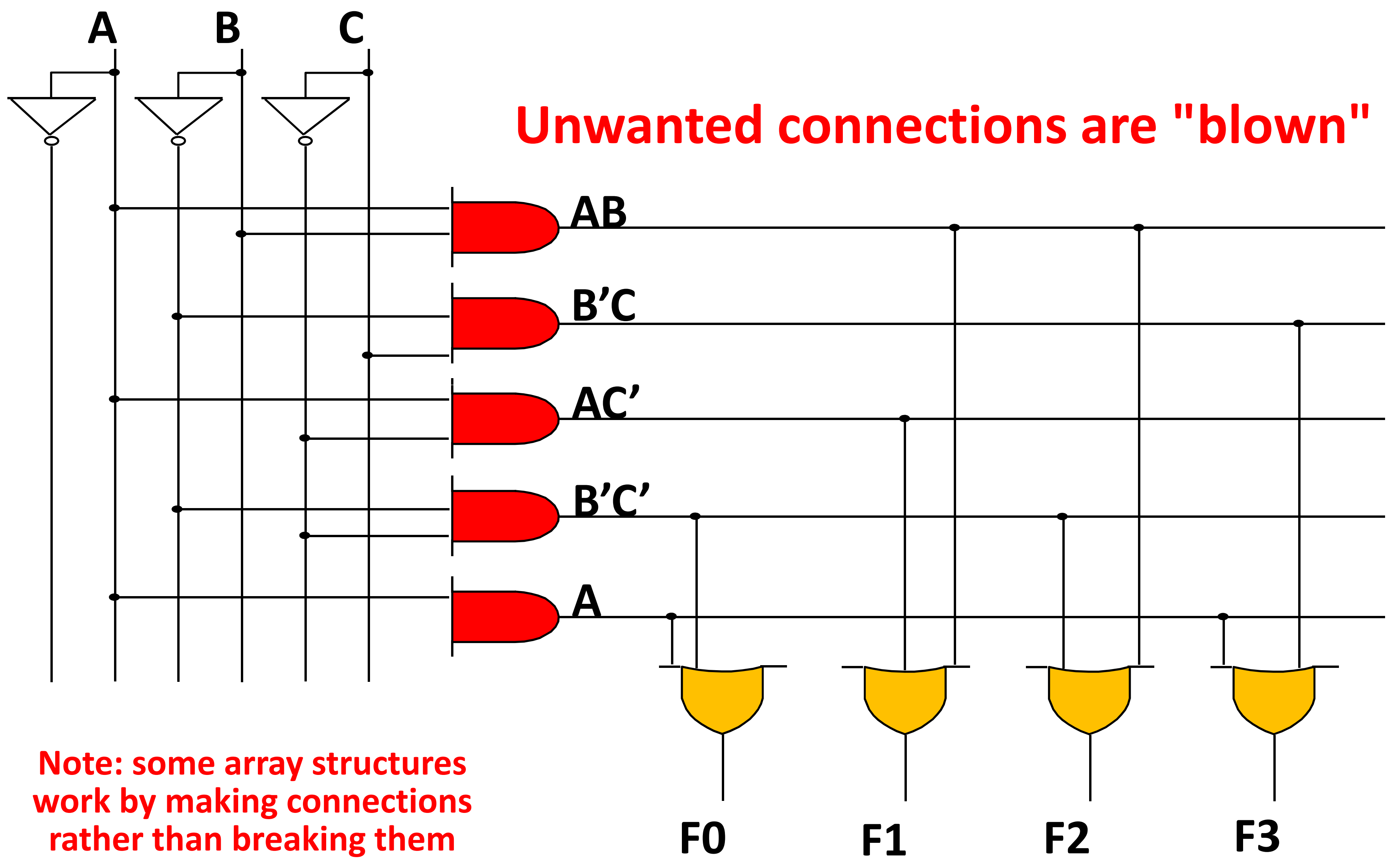
Example Continued - Unprogrammed device

All possible connections are available before programming



PLA Logic Implementation

Example Continued - Programmed part

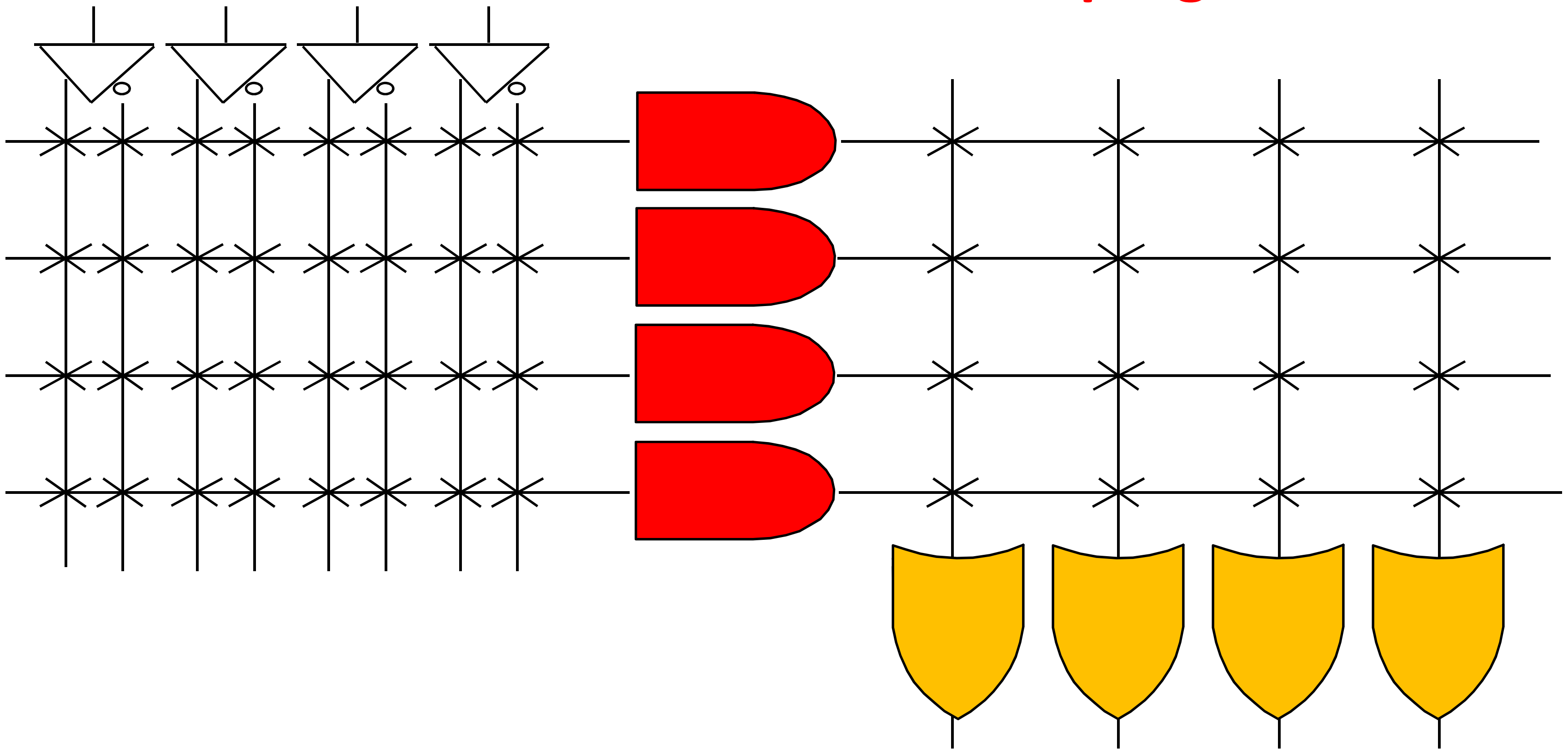


Note: some array structures work by making connections rather than breaking them

PLA Logic Implementation

Alternative representation

Un-programmed device



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

X at junction indicates a connection

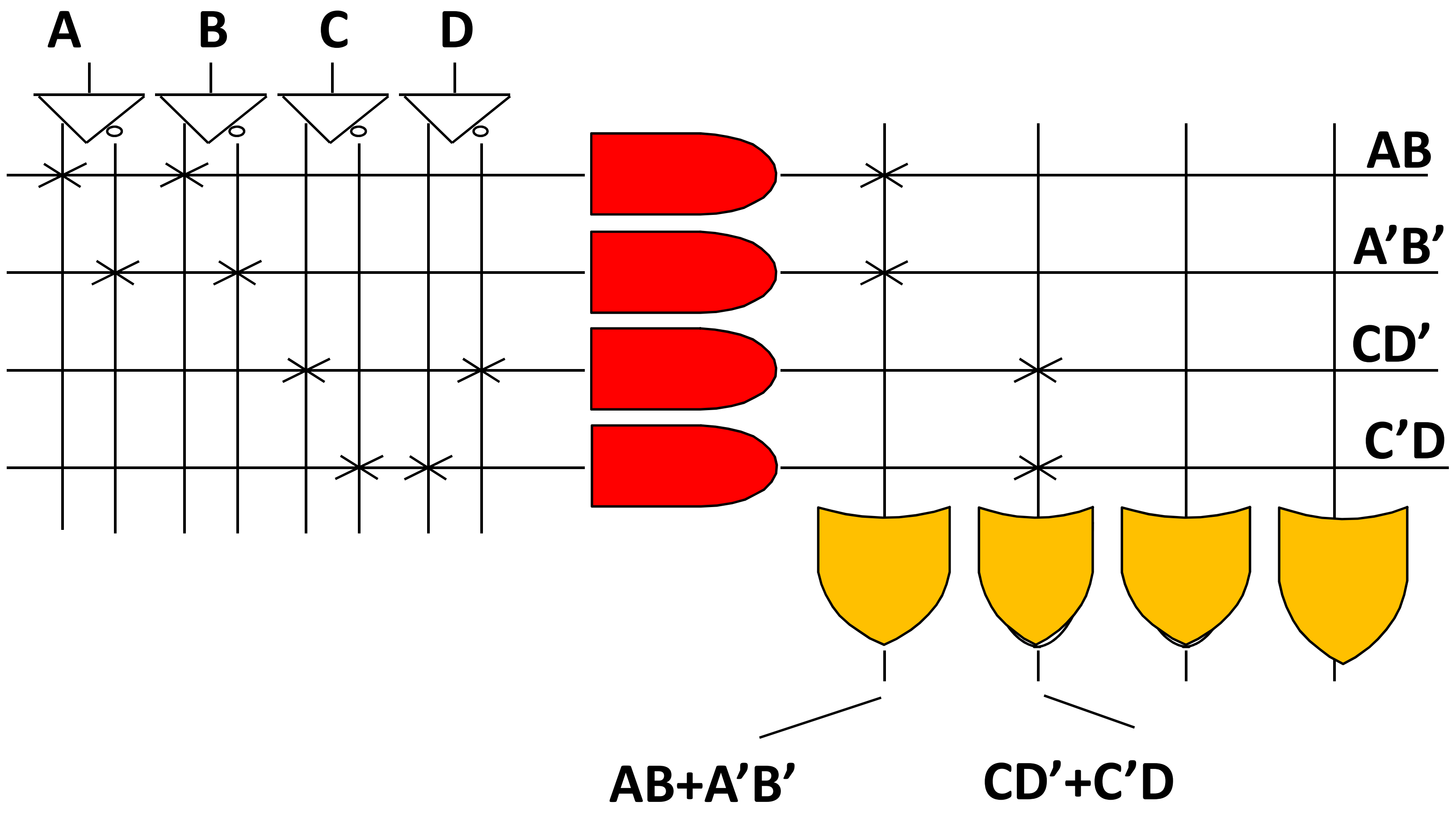
PLA Logic Implementation

Notation for implementing

F0 = A B + A' B'

F1 = CD' + C'D

Programmed device



PLA Logic Implementation

Multiple functions of A, B, C : List of all product terms

Design Example

F1 = A B C

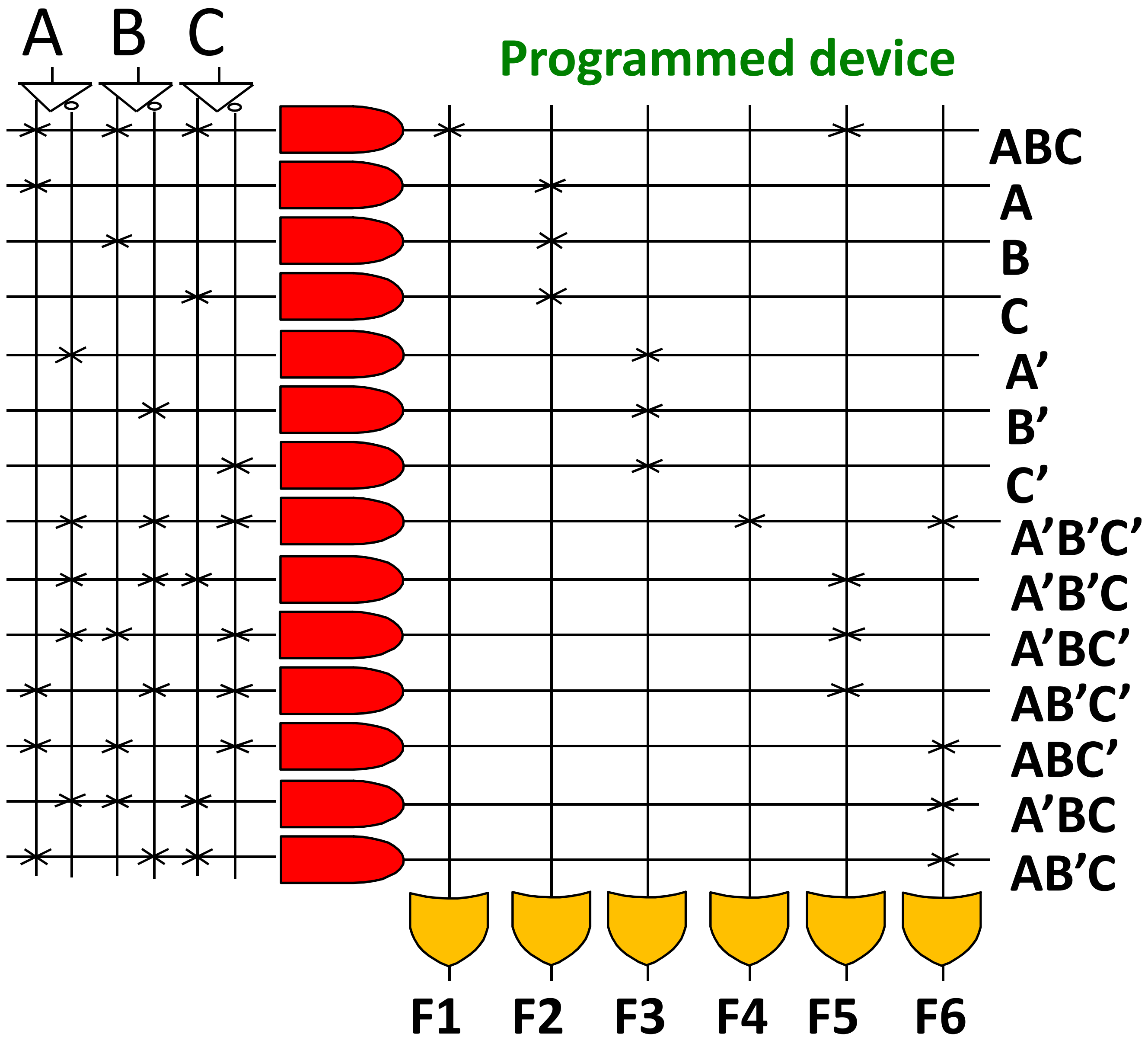
F2 = A + B + C

F3 = (A B C)'

F4 = (A + B + C)'

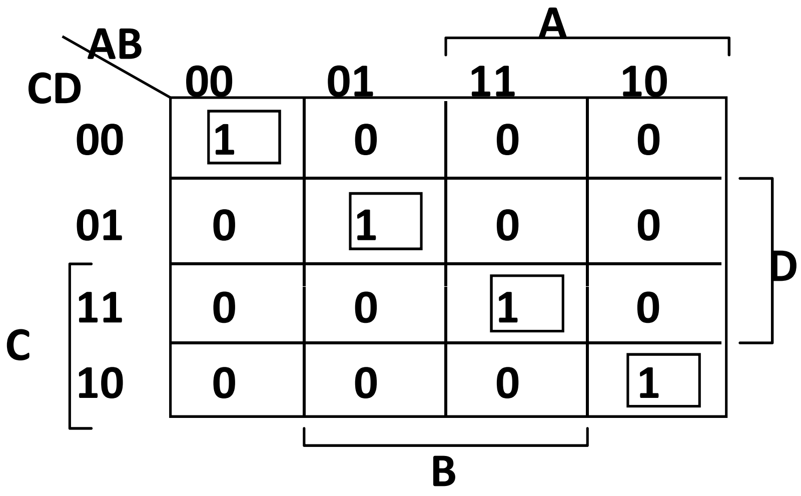
F5 = A ⊕ B ⊕ C

F6 = (A ⊕ B ⊕ C)'



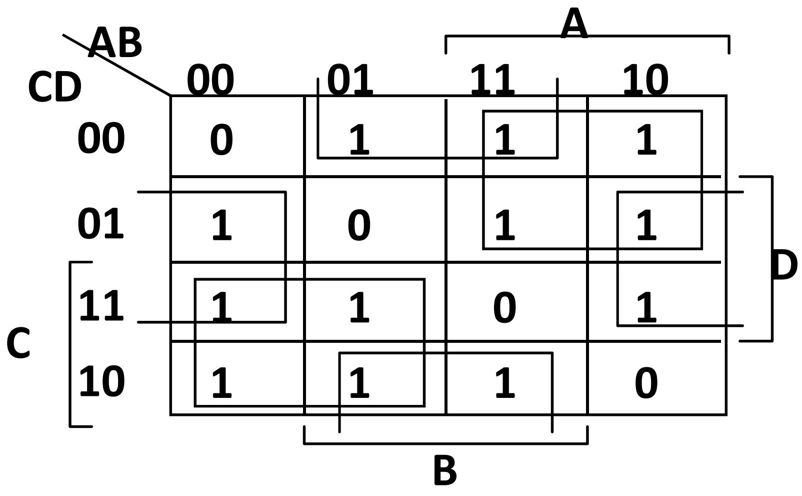
PLA Logic Implementation

Another Example: Magnitude Comparator



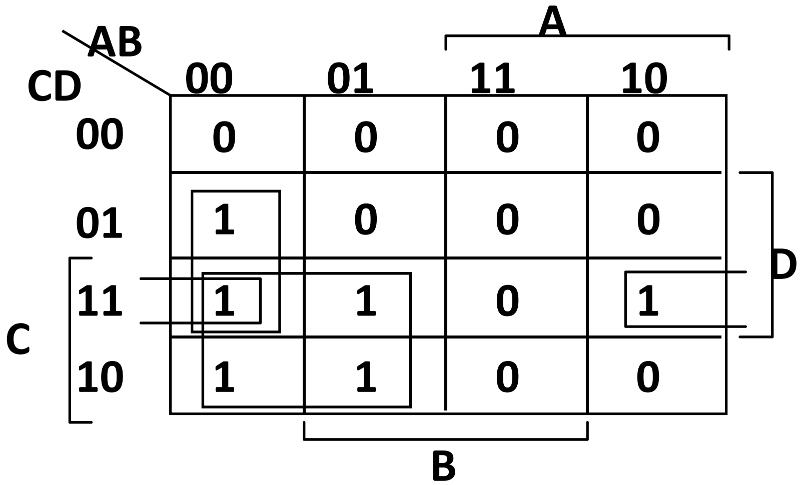
CD \ AB	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

K-map for EQ



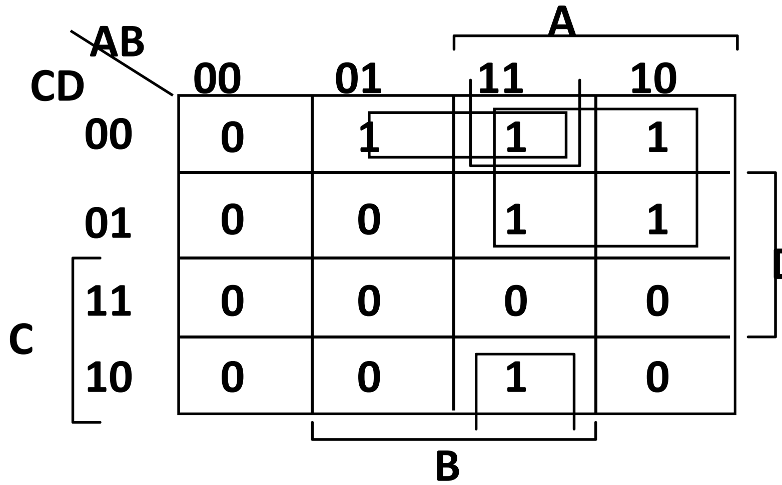
CD \ AB	00	01	11	10
00	0	1	1	1
01	1	0	1	1
11	1	1	0	1
10	1	1	1	0

K-map for NE



CD \ AB	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

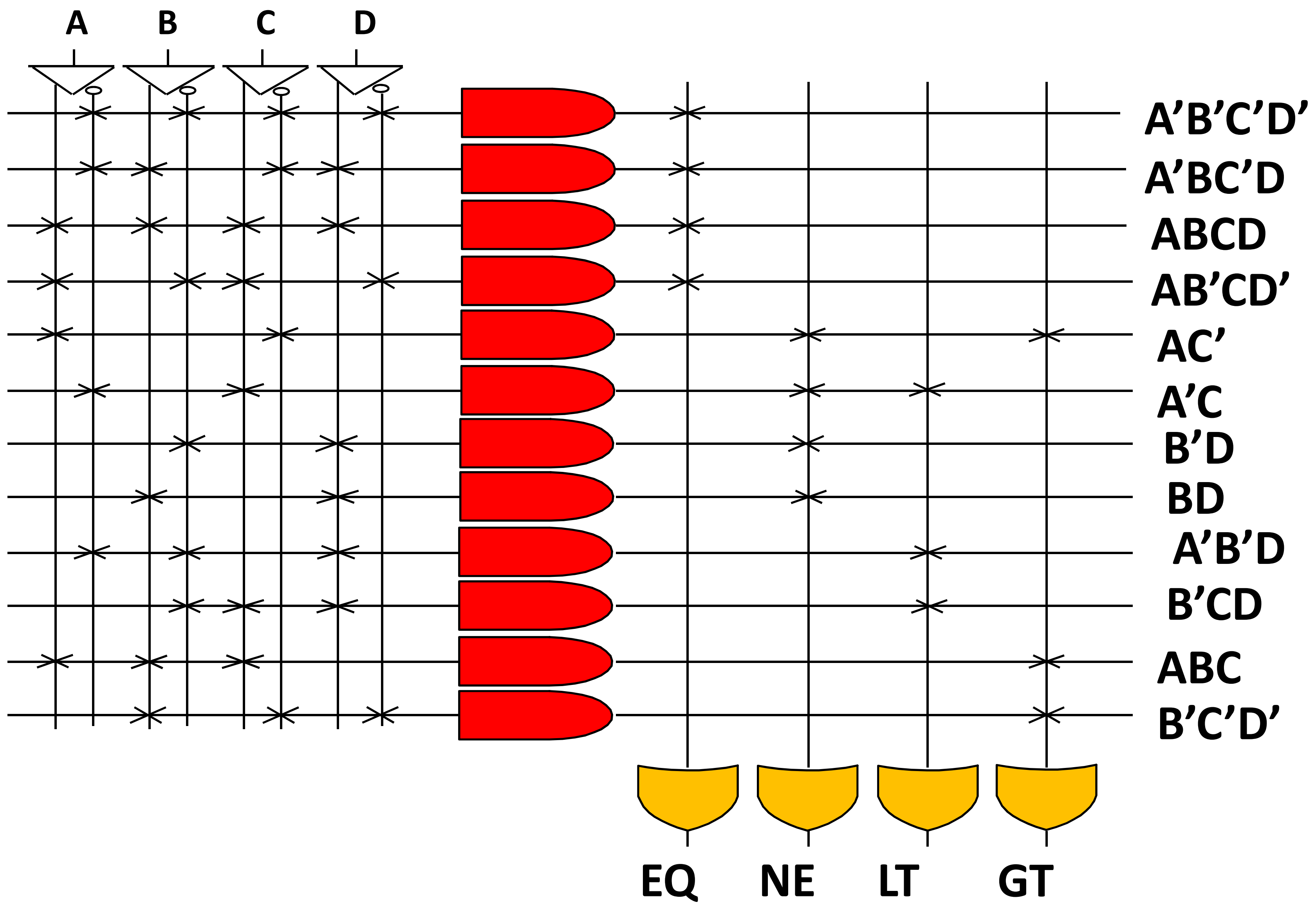
K-map for LT



CD \ AB	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

K-map for GT

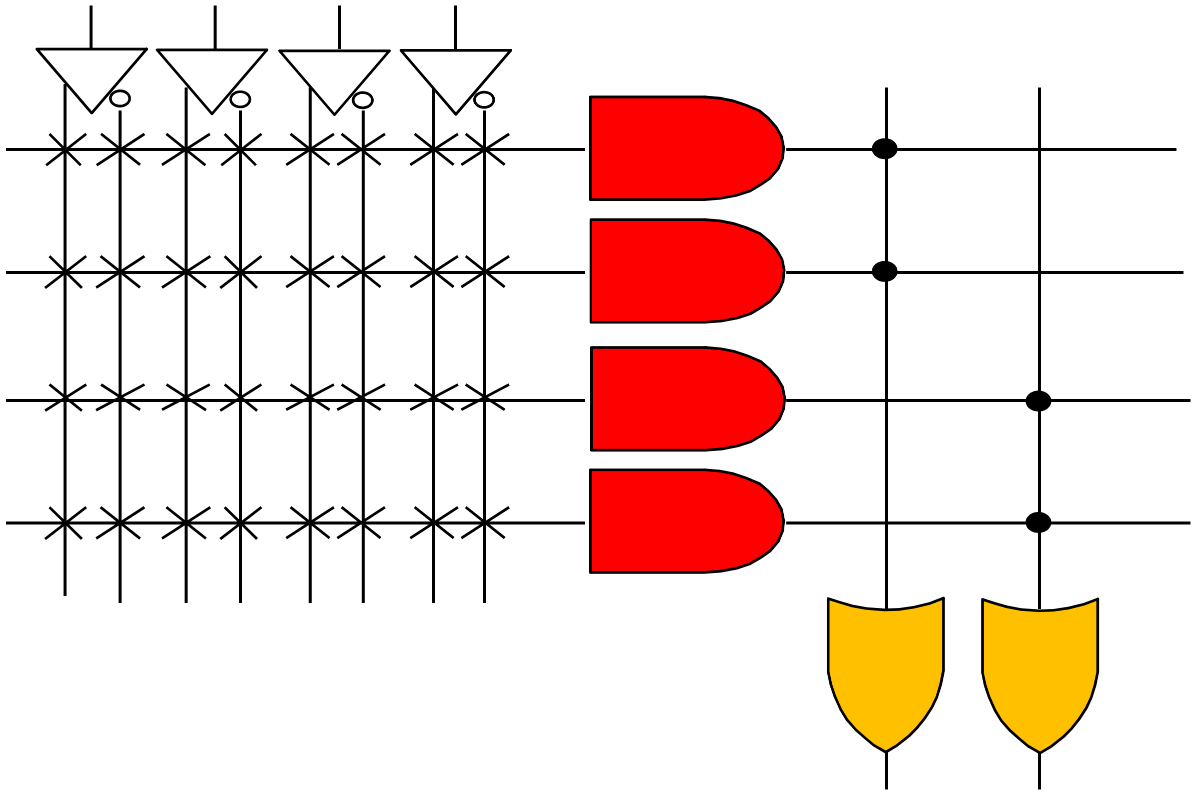
PLA Logic Imp: Magnitude Comparator



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

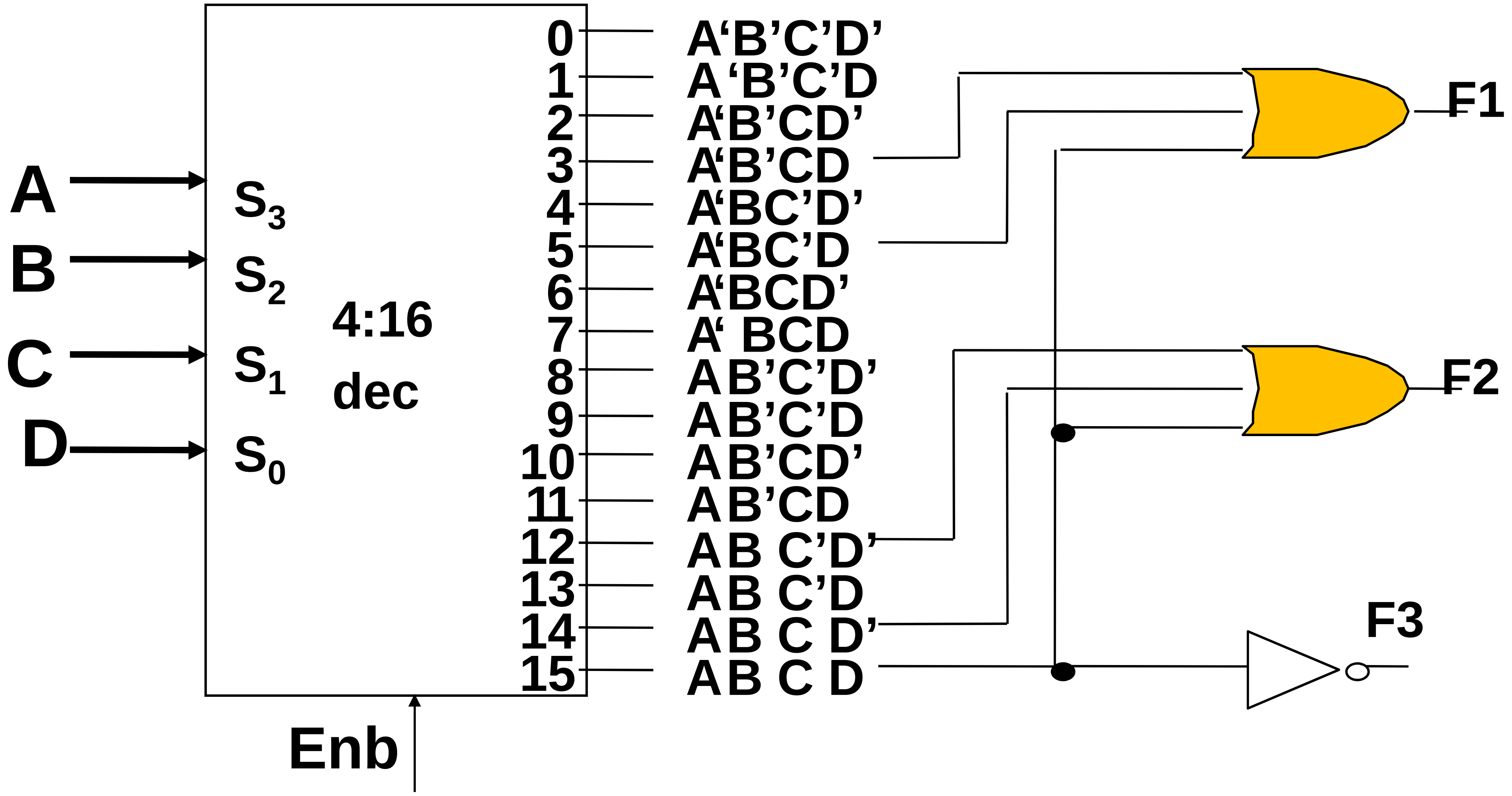
- Of the two organizations the PLA is the most flexible
  - One PLA can implement a huge range of logic functions
  - BUT many pins; large package, higher cost
- PALs are more restricted / you trade number of OR terms vs number of outputs
  - Many device variations needed
  - **Each device is cheaper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

Read-Only Memory

ROM

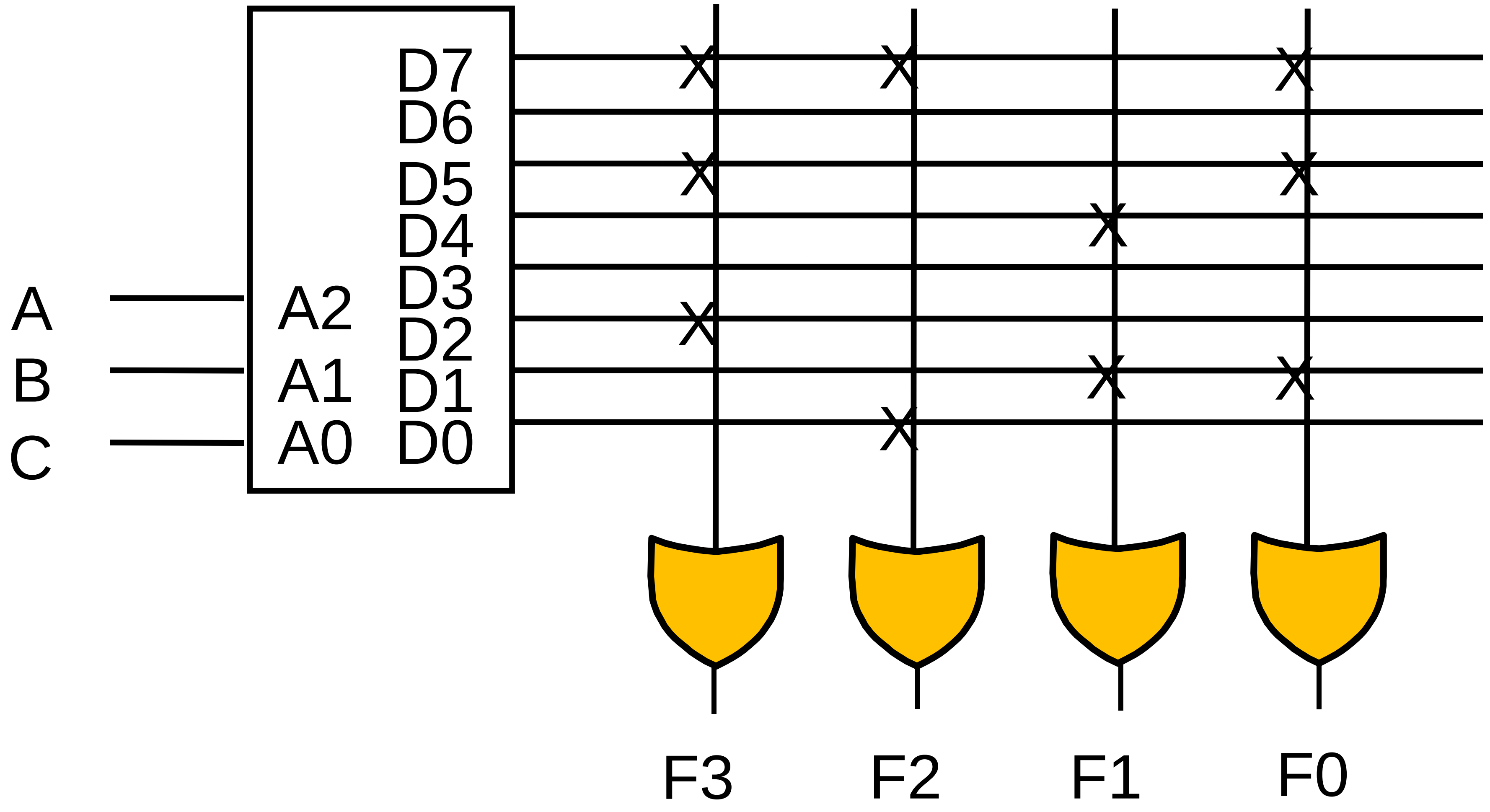
ROM

- **Decoder : Produces minterms**
- **Ors : Produce SOP's**

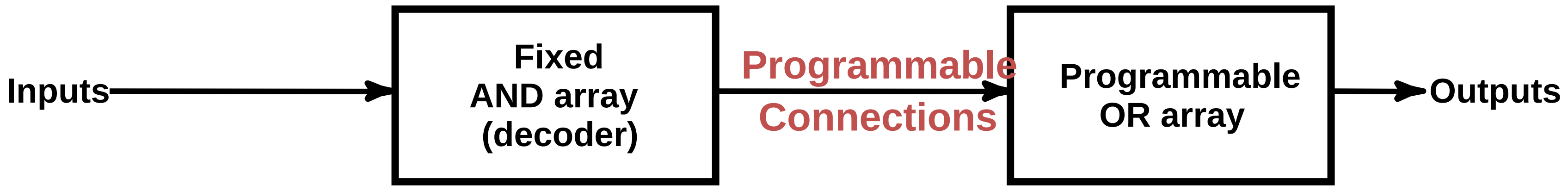


ROM

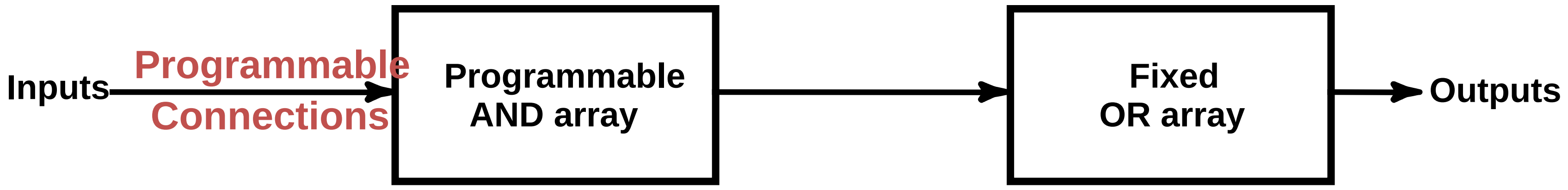
- **A decoder**
- **A set of programmable OR's**



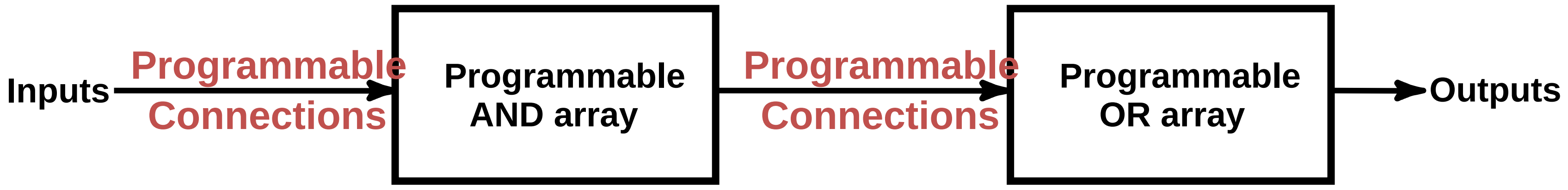
ROM vs. PLA/PAL



(a) Programmable read-only memory (PROM)



(b) Programmable array logic (PAL) device



(c) Programmable logic array (PLA) device

General Logic Implementation

- Given a 2^kxn ROM, we can implement ANY combinational circuit with at most k inputs and at most n outputs.
- Why?
  - k-to-2^k decoder will generate all 2^k possible minterms
  - Each of the OR gates must implement a $\sum m()$
  - Each $\sum m()$ can be programmed

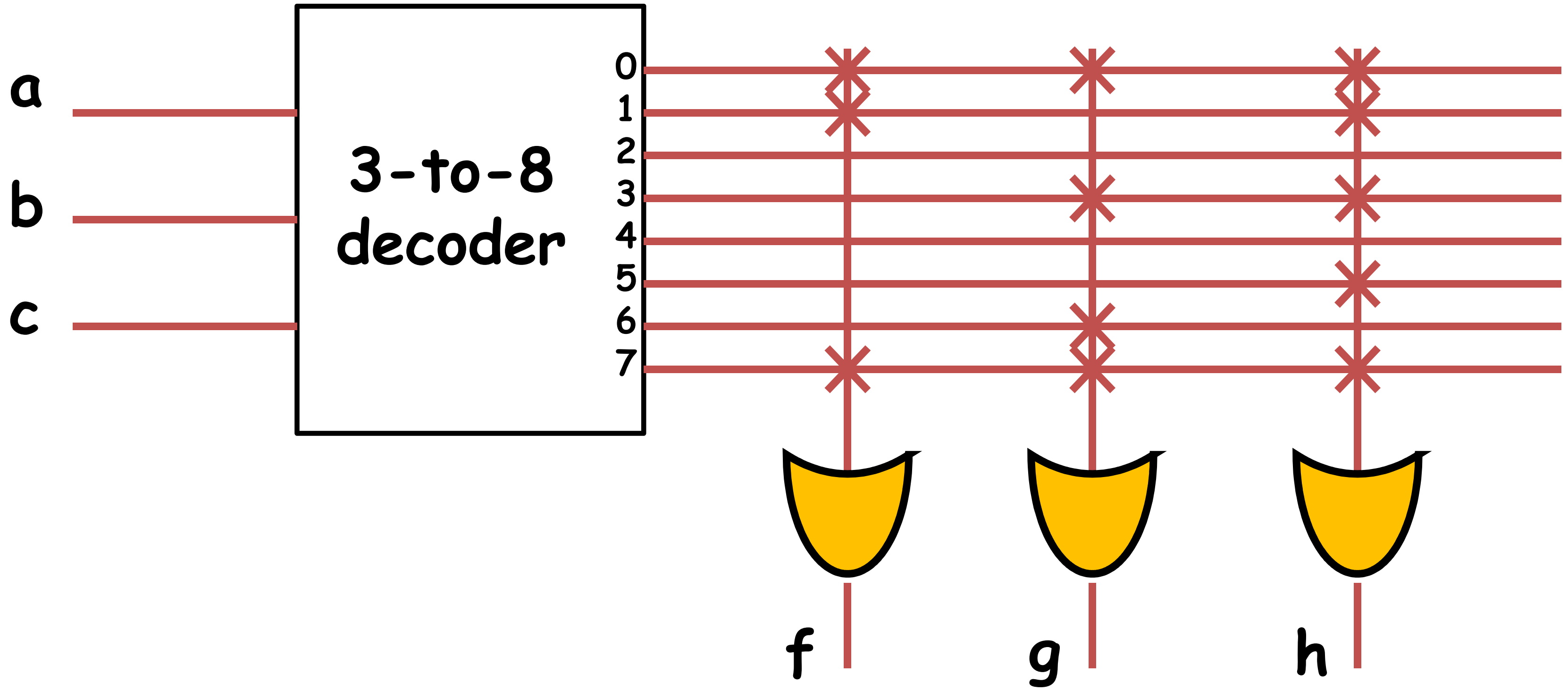
Example

- Find a ROM-based circuit implementation for:
  - f(a,b,c) = a’b’ + abc
  - g(a,b,c) = a’b’c’ + ab + bc
  - h(a,b,c) = a’b’ + c
- Solution:
  - Express f(), g(), and h() in $\sum$m() format (use truth tables)
  - Program the ROM based on the 3 $\sum$m()’s

Example

- There are 3 inputs and 3 outputs, thus we need a 8x3 ROM block.

$f = \sum m(0, 1, 7)$, $g = \sum m(0, 3, 6, 7)$, $h = \sum m(0, 1, 3, 5, 7)$



ROM as a Memory

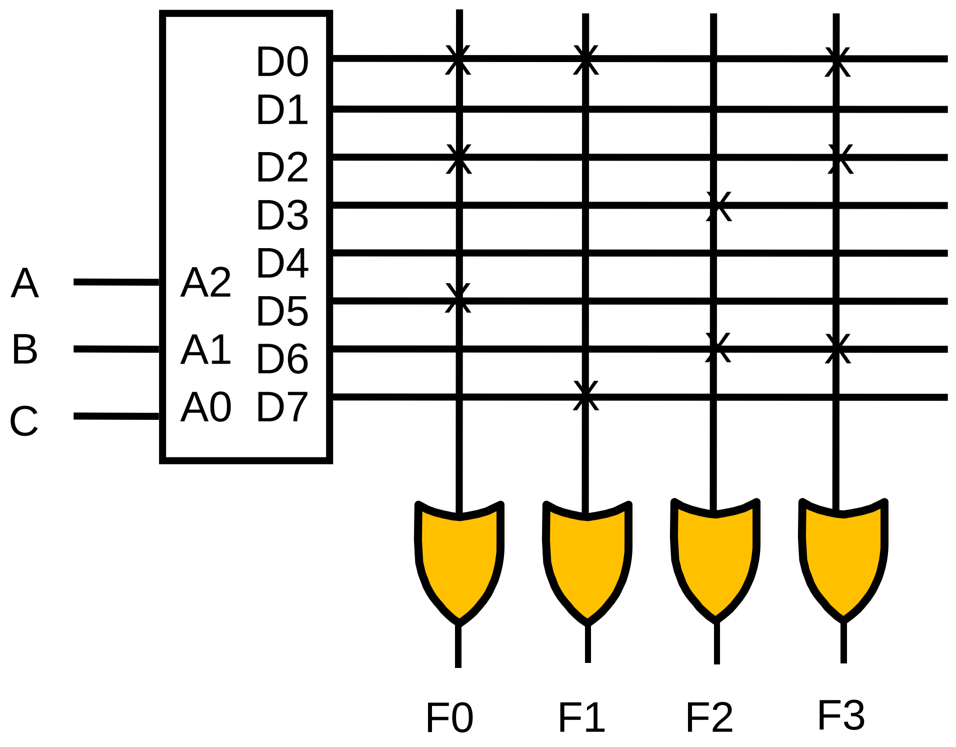
- Read Only Memories (ROM) or Programmable Read Only Memories (PROM) have:
  - N input lines,
  - M output lines, and
  - 2^N decoded minterms.
- Can be viewed as a memory with the inputs as addresses of data (output values),
  - hence ROM or PROM names!

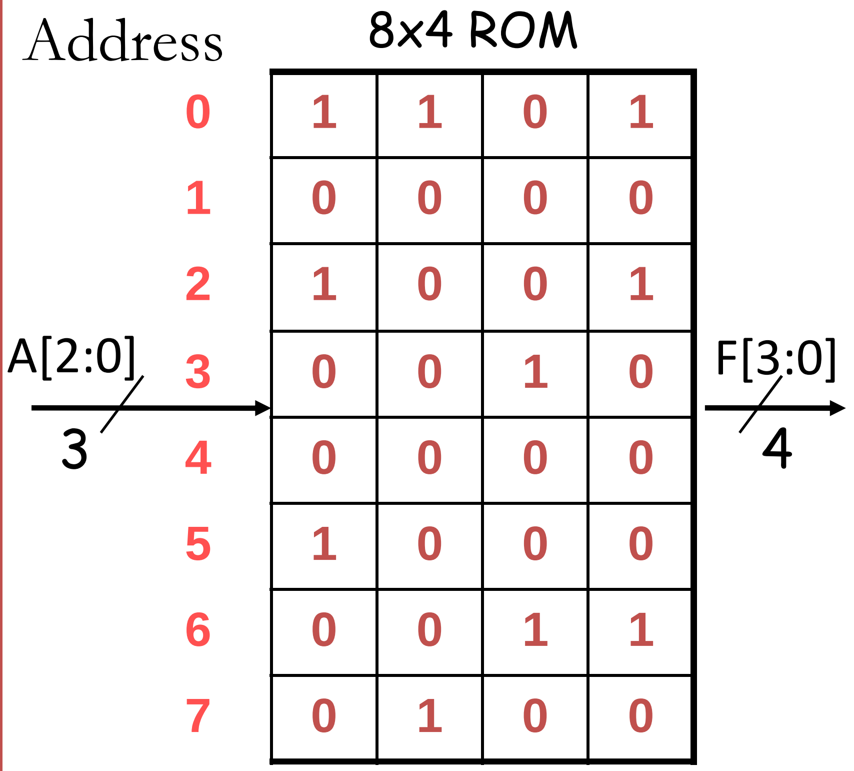
Memories

- Volatile: Random Access Memory (RAM)
  - SRAM "static"
  - DRAM "dynamic"
- Non-Volatile: Read Only Memory (ROM):
  - Mask ROM "mask programmable"
  - EPROM "electrically programmable"
  - EEPROM “electrically erasable electrically programmable"
  - FLASH memory - similar to EEPROM with programmer integrated on chip

ROM as Memory

- **Read Example**: For input $(A_2,A_1,A_0) = 011$, output is $(F_0,F_1,F_2,F_3) = 0010$.
- What are functions F_3, F_2, F_1 and F_0 in terms of (A_2, A_1, A_0)?





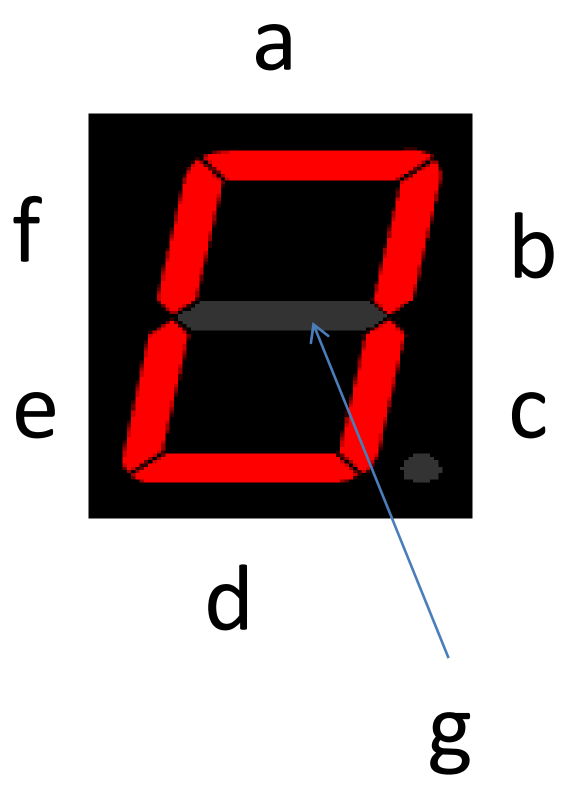
Address				
0	1	1	0	1
1	0	0	0	0
2	1	0	0	1
3	0	0	1	0
4	0	0	0	0
5	1	0	0	0
6	0	0	1	1
7	0	1	0	0

$A[2:0] = A_2A_1A_0$ $F[3:0]=F_3F_2F_1F_0$

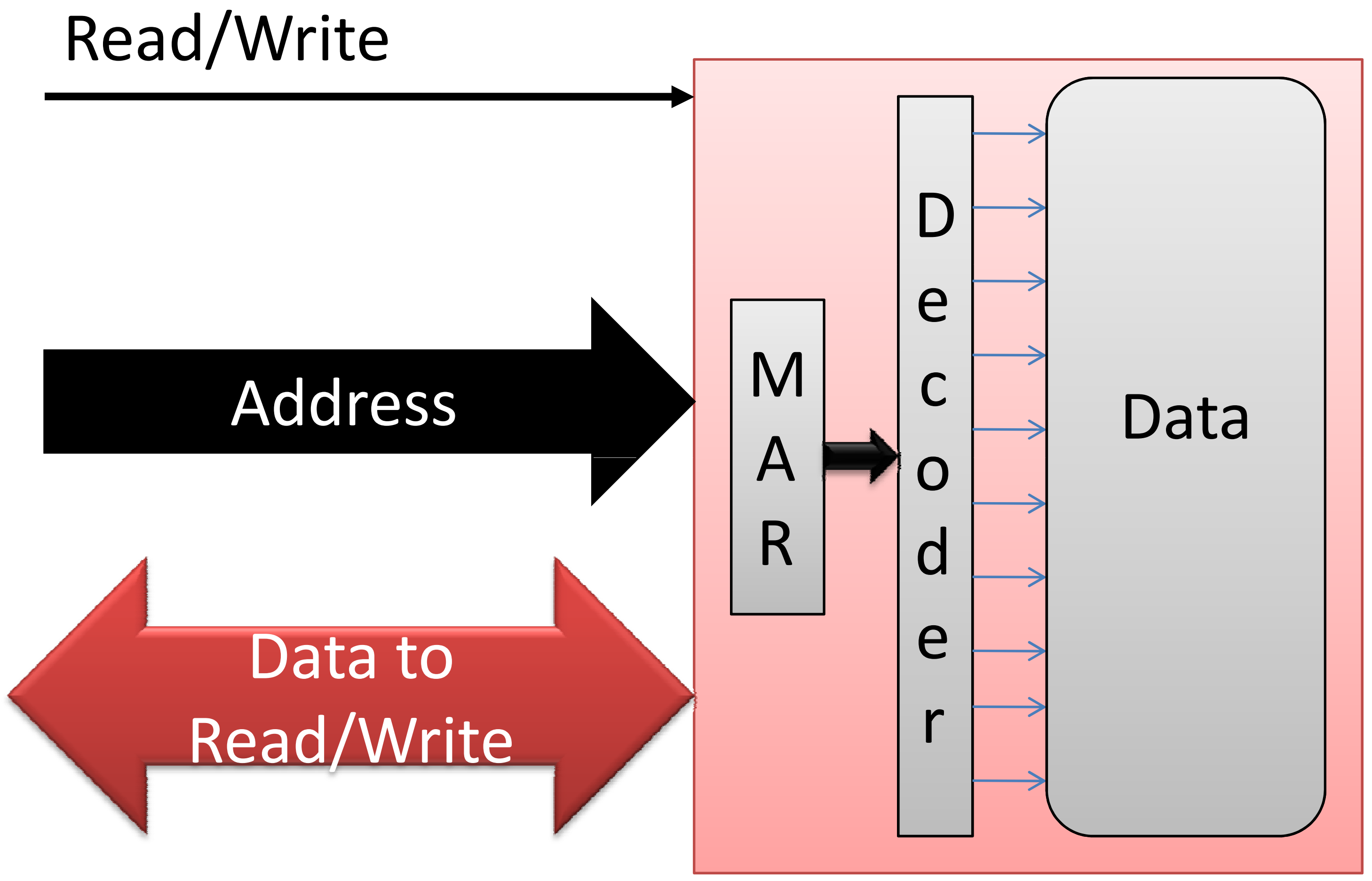
Design by ROM: Example

- **BCD to 7 Segment Display Controller**

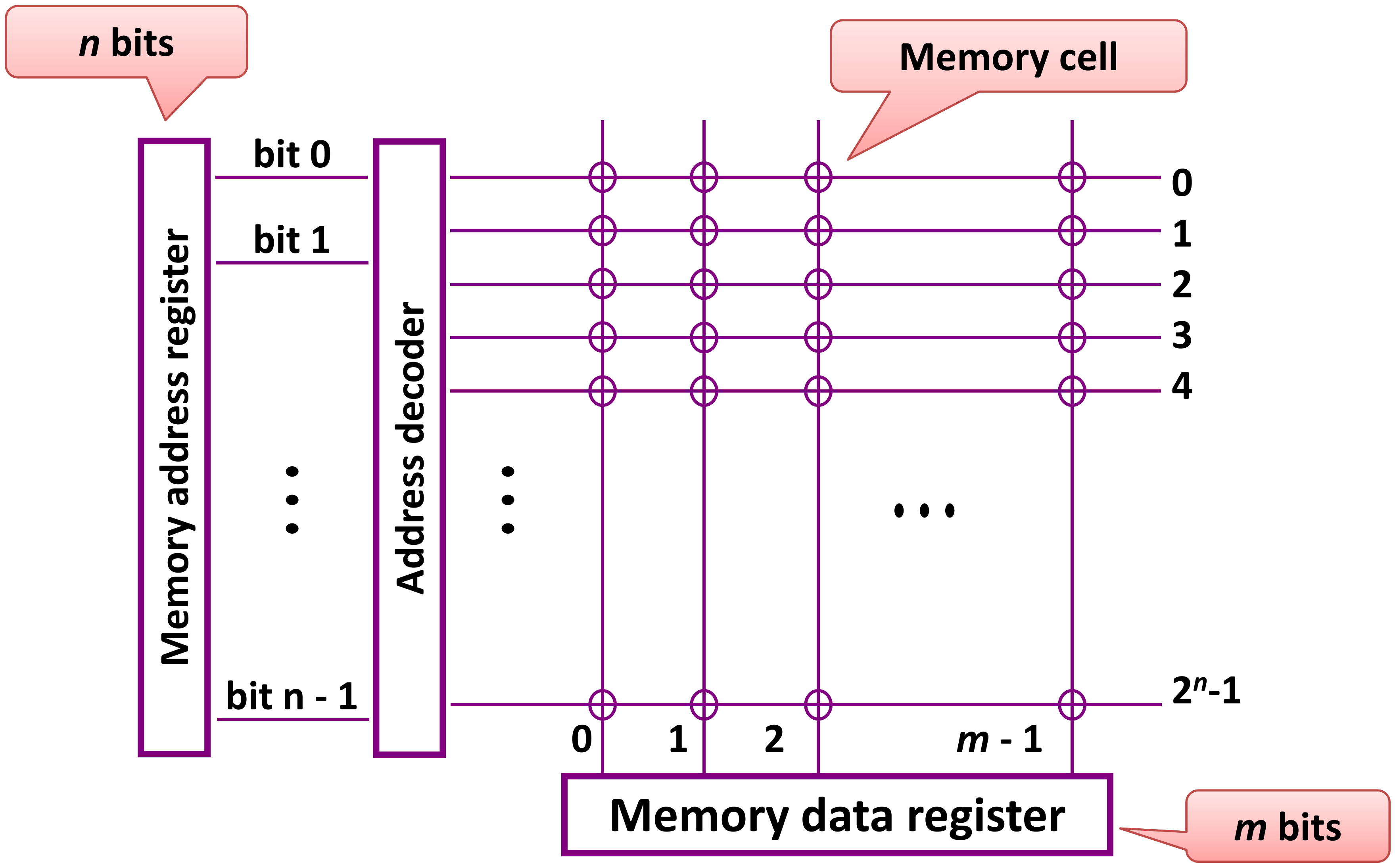
A B C D	a	b	c	d	e	f	g
0 0 0 0	1	1	1	1	1	1	0
0 0 0 1	0	1	1	0	0	0	0
0 0 1 0	1	1	0	1	1	0	1
0 0 1 1	1	1	1	1	0	0	1
0 1 0 0	0	1	1	0	0	1	1
0 1 0 1	1	0	1	1	0	1	1
0 1 1 0	1	0	1	1	1	1	1
0 1 1 1	1	1	1	0	0	0	0
1 0 0 0	1	1	1	1	1	1	1
1 0 0 1	1	1	1	0	0	1	1
1 0 1 0	X	X	X	X	X	X	X
1 0 1 1	X	X	X	X	X	X	X
1 1 0 0	X	X	X	X	X	X	X
1 1 0 1	X	X	X	X	X	X	X
1 1 1 0	X	X	X	X	X	X	X
0 1 1 1	X	X	X	X	X	X	X



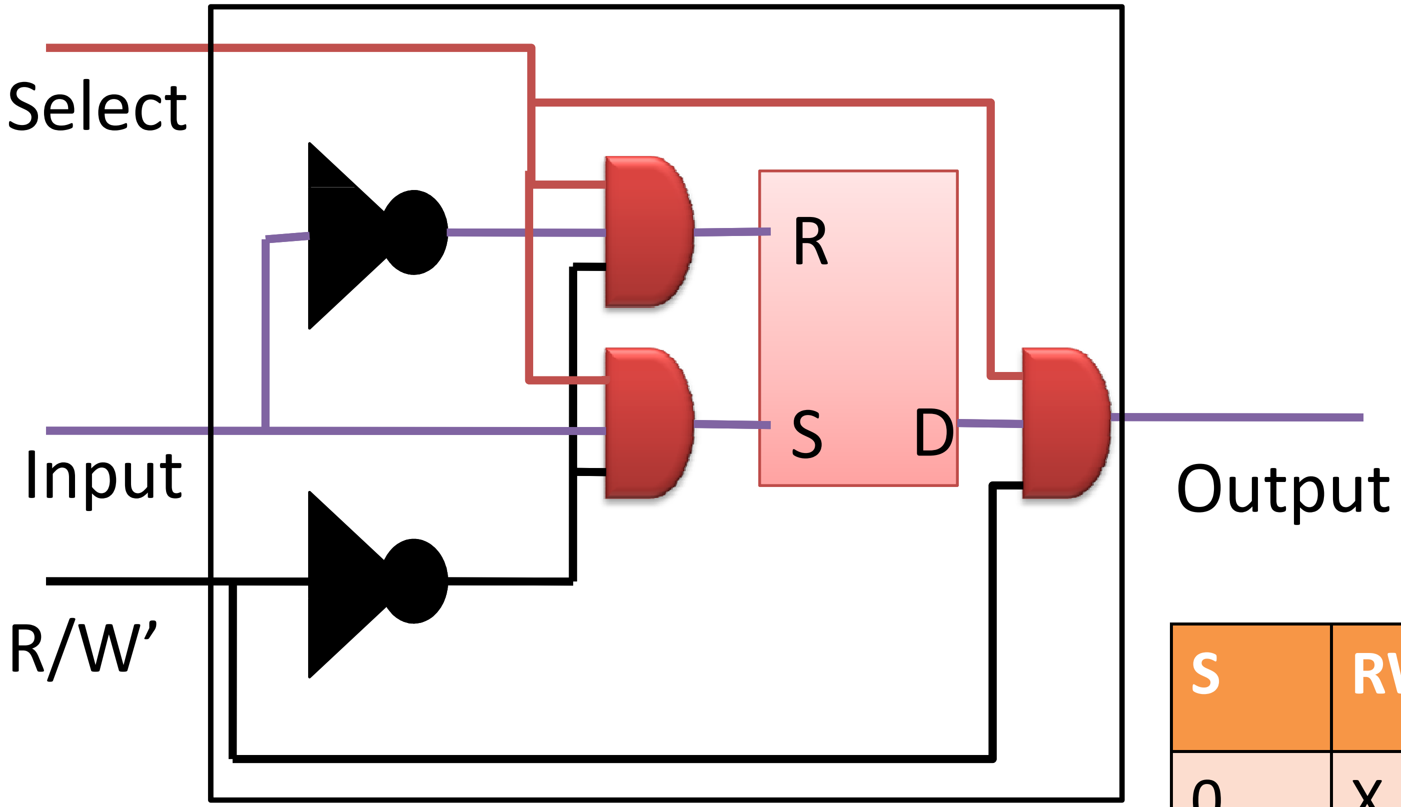
Memory Unit



Memory Unit

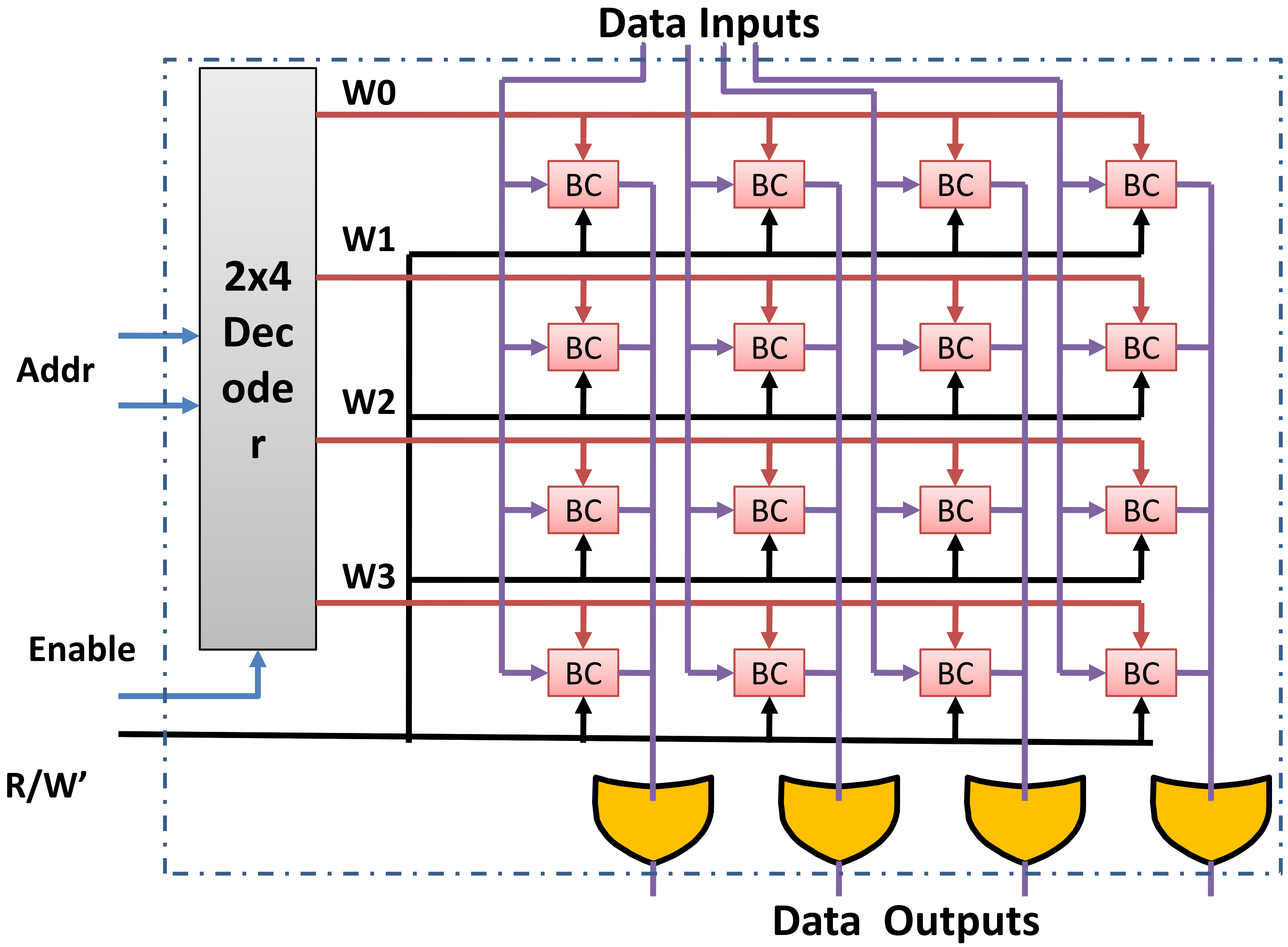

n bits
Memory cell
Memory address register
bit 0
bit 1
bit n - 1
Address decoder
0
1
2
3
4
2^n-1
0
1
2
m - 1
Memory data register
m bits

Memory Cell



S	RW'	D	O/p
0	X	X	0
1	1	X	D
1	0	In	0

Memory


Data Inputs
W0
W1
W2
W3
2x4 Decoder
Addr
Enable
R/W’
BC
Data Outputs