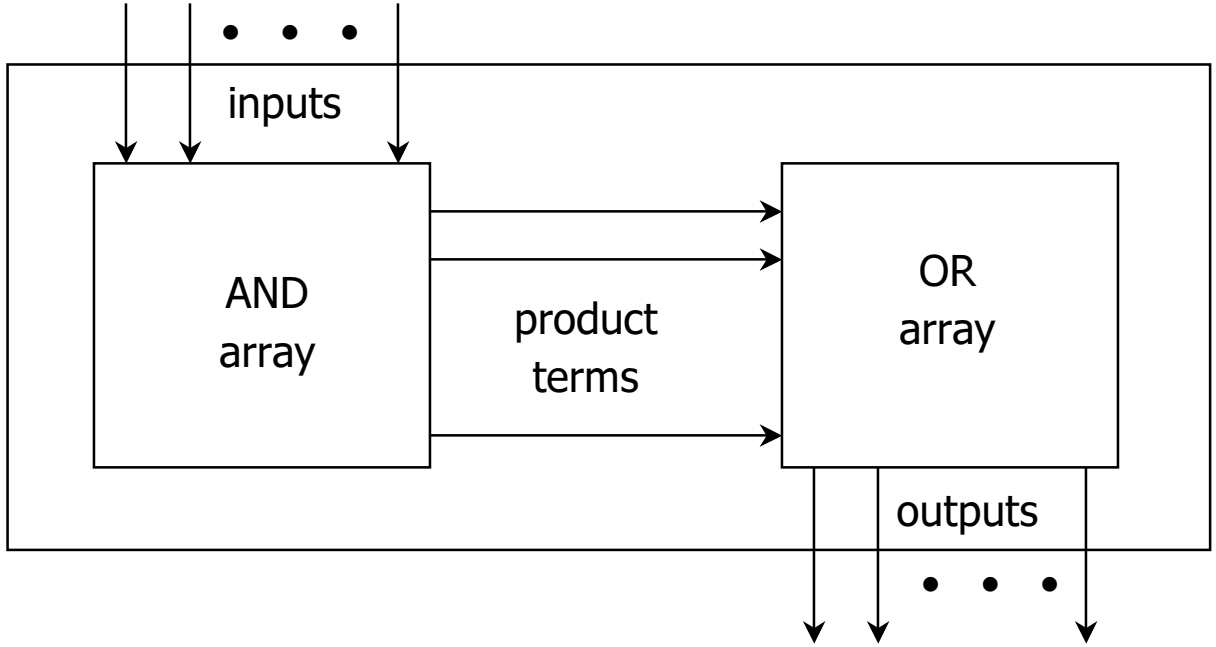
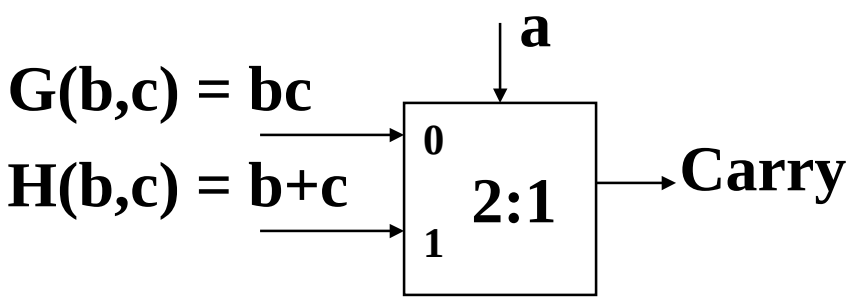
Programmable logic arrays (PLA)

- Pre-fabricated building block of many AND/OR gates
  - actually NOR or NAND
  - "personalized" by making or breaking connections among the gates
  - programmable array block diagram for sum of products form

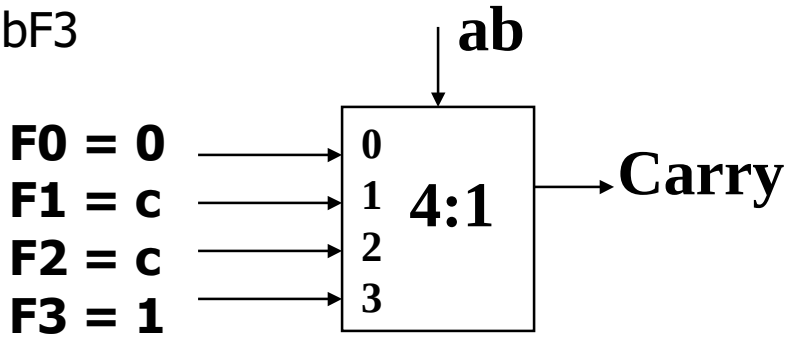


Multiplexor Logic

- F(a,b,c) = a'F(0,b,c) + aF(1,b,c)
  - Let G(b,c) = F(0,b,c)
  - Let H(b,c) = F(1,b,c)



  - Example FullAdder Carry = ab + ac + bc
  - G = bc H = b + c + bc = b + c
- F(a,b,c) = a'b'F(0,0,c) + a'bF(0,1,c) + ab'F(1,0,c) + abF(1,1,c)
  - F(a,b,c) = a'b'F0 + a'bF1 + ab'F2 or abF3



Programmable Logic Devices

- Shared product terms among outputs

example:

F0 = A + B' C'
F1 = A C' + A B
F2 = B' C' + A B
F3 = B' C + A

input side:

1 = uncomplemented in term
0 = complemented in term
– = does not participate

output side:

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

personality matrix

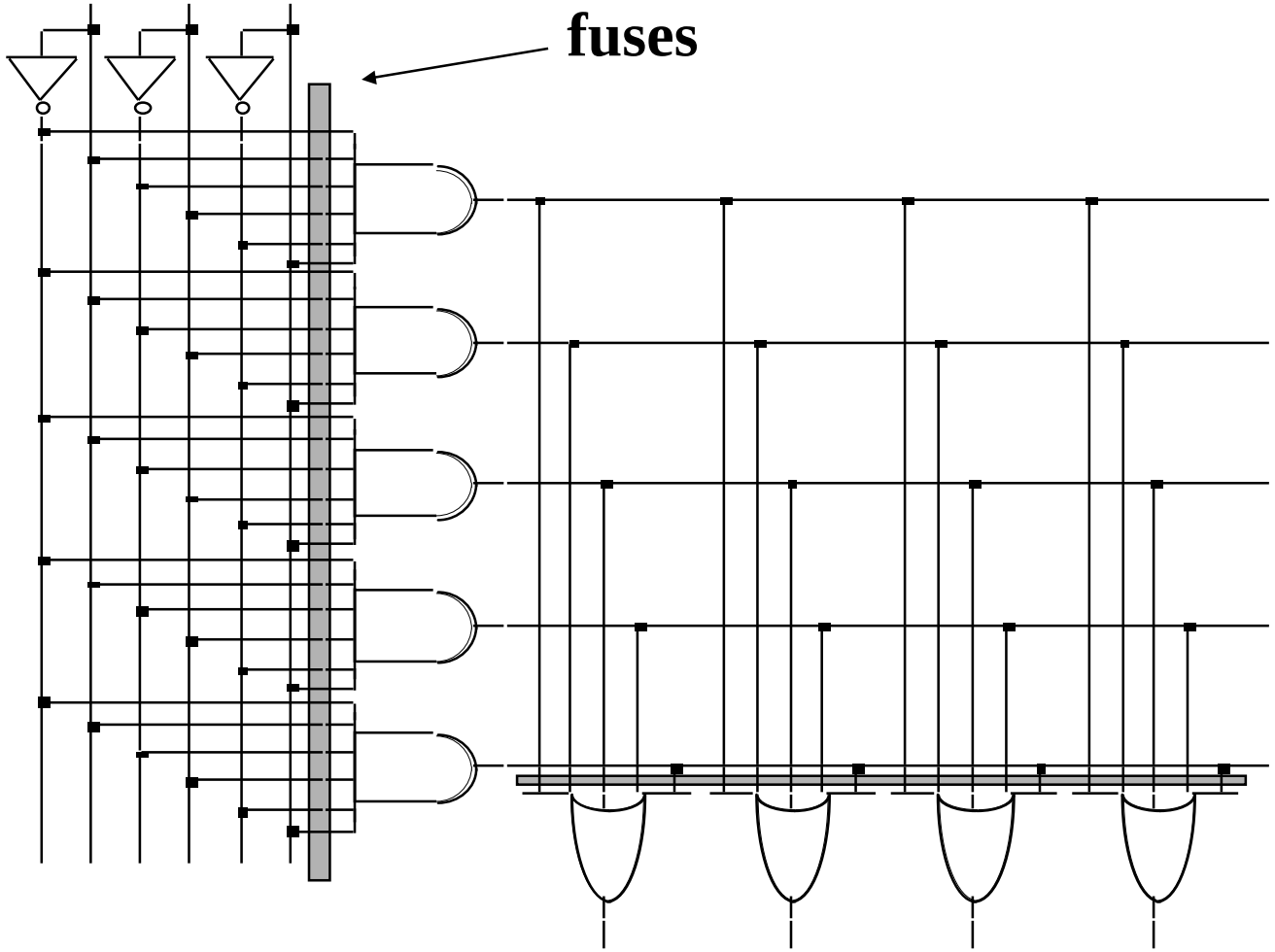
product term	inputs A	B	C	outputs F0	F1	F2	F3
AB	1	1	–	–	1	1	–
B'C	–	0	1	–	–	–	1
AC'	1	–	0	–	1	–	–
B'C'	–	0	0	1	–	1	–
A	1	–	–	1	–	0	1

and rows

or columns (note common subexpression re-use)

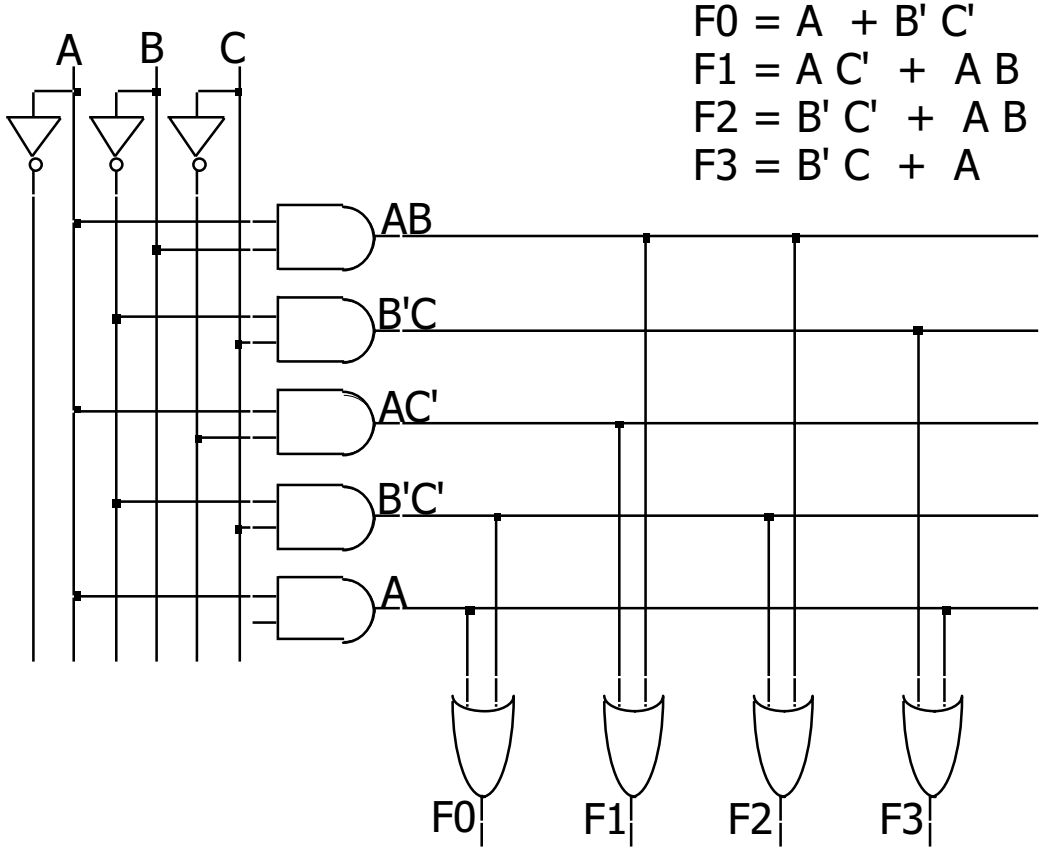
Before programming

- All possible connections are available before "programming"
  - in reality, all AND and OR gates are NANDs



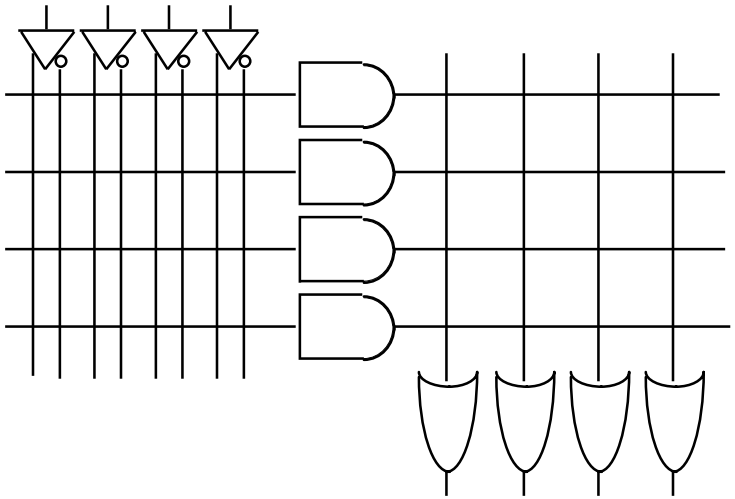
After programming

- Unwanted connections are "blown"
  - fuse (normally connected, break unwanted ones)
  - anti-fuse (normally disconnected, make wanted connections)



Alternate representation for high fan-in structures

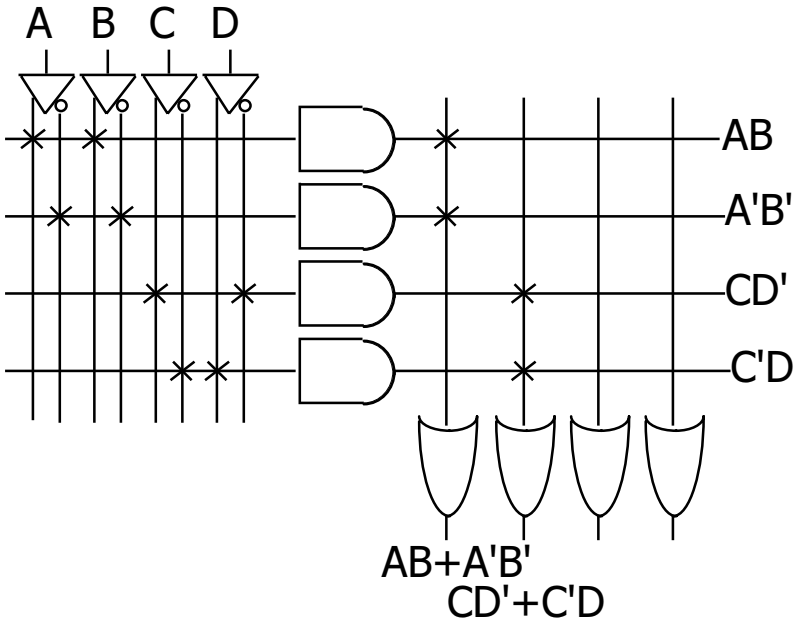
- Short-hand notation so we don't have to draw all the wires
  - × signifies a connection is present and perpendicular signal is an input to gate



notation for implementing

F0 = A B + A' B'

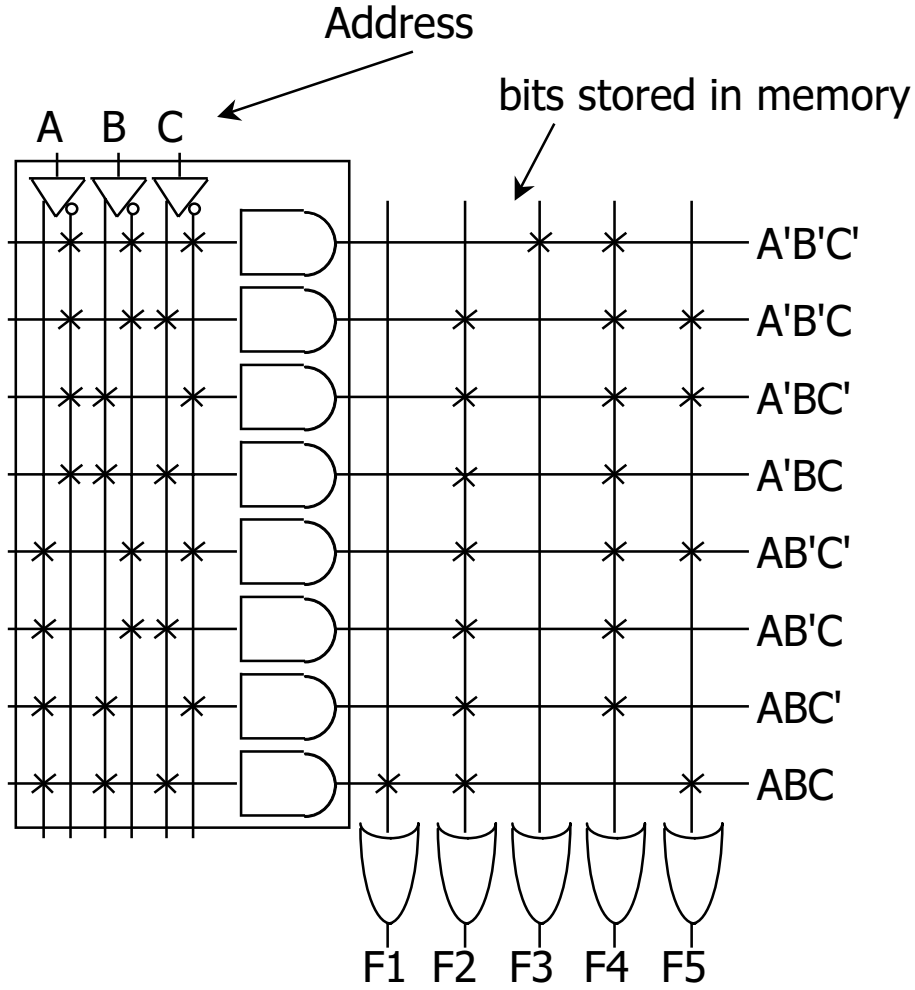
F1 = C D' + C' D



PLA as ROM

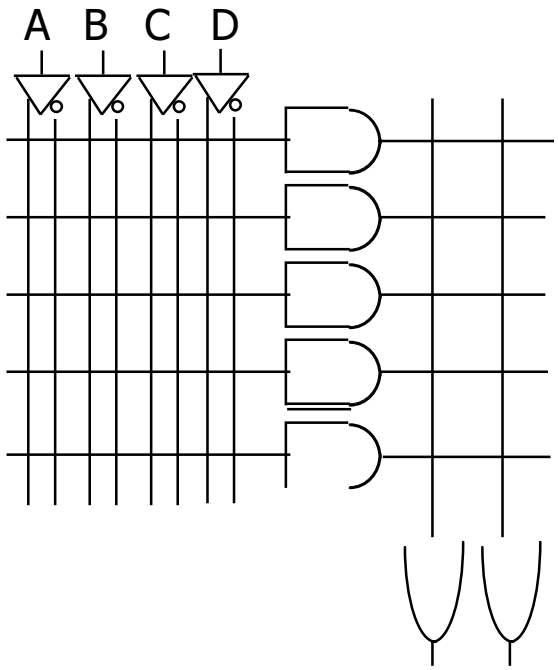
- Multiple functions of A, B, C
  - F1 = A B C
  - F2 = A + B + C
  - F3 = A' B' C'
  - F4 = A' + B' + C'
  - F5 = A xor B xor C

A	B	C	F1	F2	F3	F4	F5
0	0	0	0	0	1	1	0
0	0	1	0	1	0	1	1
0	1	0	0	1	0	1	1
0	1	1	0	1	0	1	0
1	0	0	0	1	0	1	1
1	0	1	0	1	0	1	0
1	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1



Common Sub-Expression Extraction and Use

Implemement
F = Σm(5,7,10,14,15)
G =Σm(6,7,9,13,15)

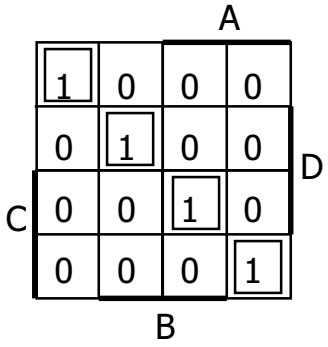


0	1	3	2
4	5 **F**	7 **F**	6
12	13	15 **F**	14 **F**
8	8	11	10 **F**

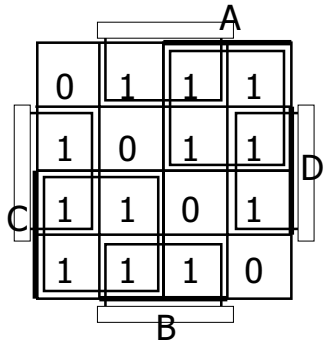
0	1	3	2
4	5	7 **G**	6 **G**
12	13 **G**	15 **G**	14
8	8 **G**	11	10

PALs and PLAs: another design example

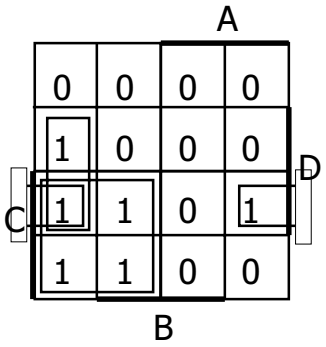
- Magnitude comparator



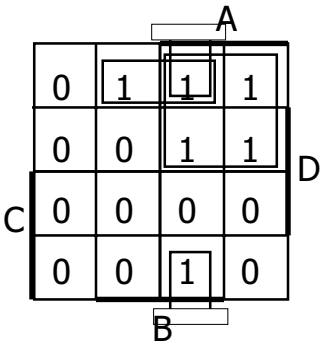
K-map for EQ



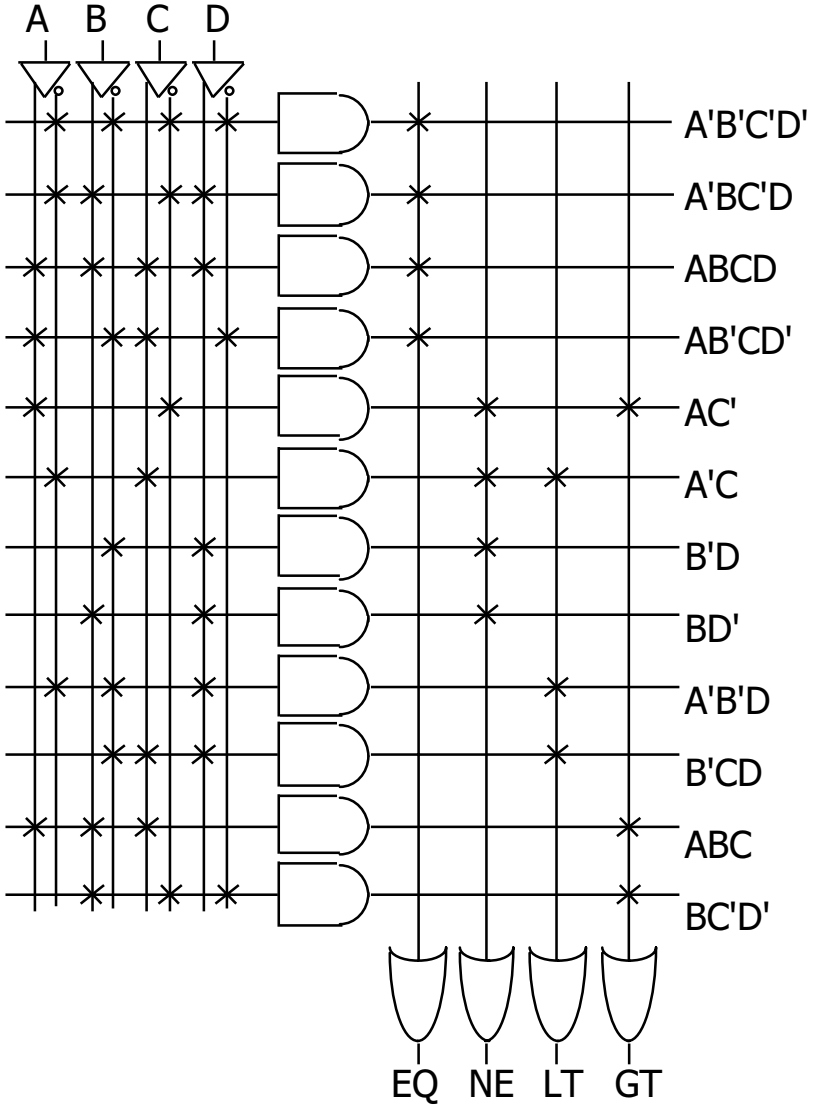
K-map for NE



K-map for LT



K-map for GT

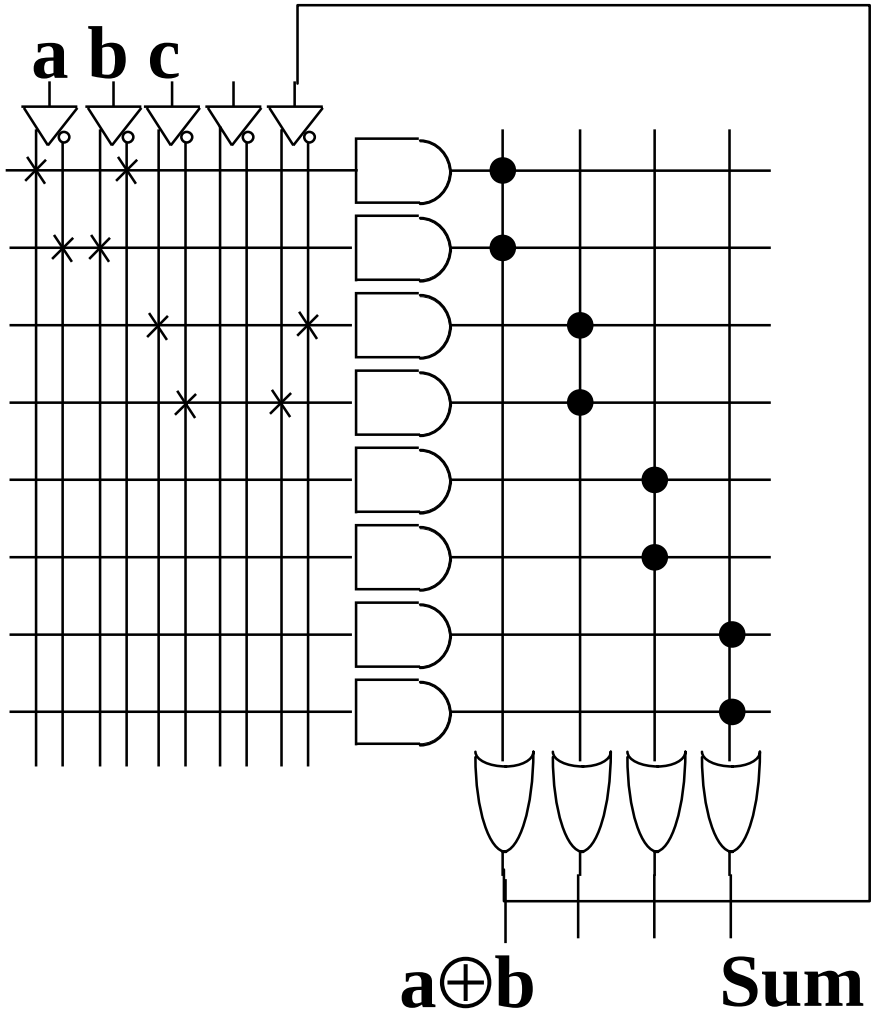


PALs and PLAs

- Programmable logic array (PLA)
  - what we've seen so far
  - unconstrained fully-general AND and OR arrays
- Programmable array logic (PAL)
  - Fixed OR array
  - faster and smaller OR plane
  - No term sharing

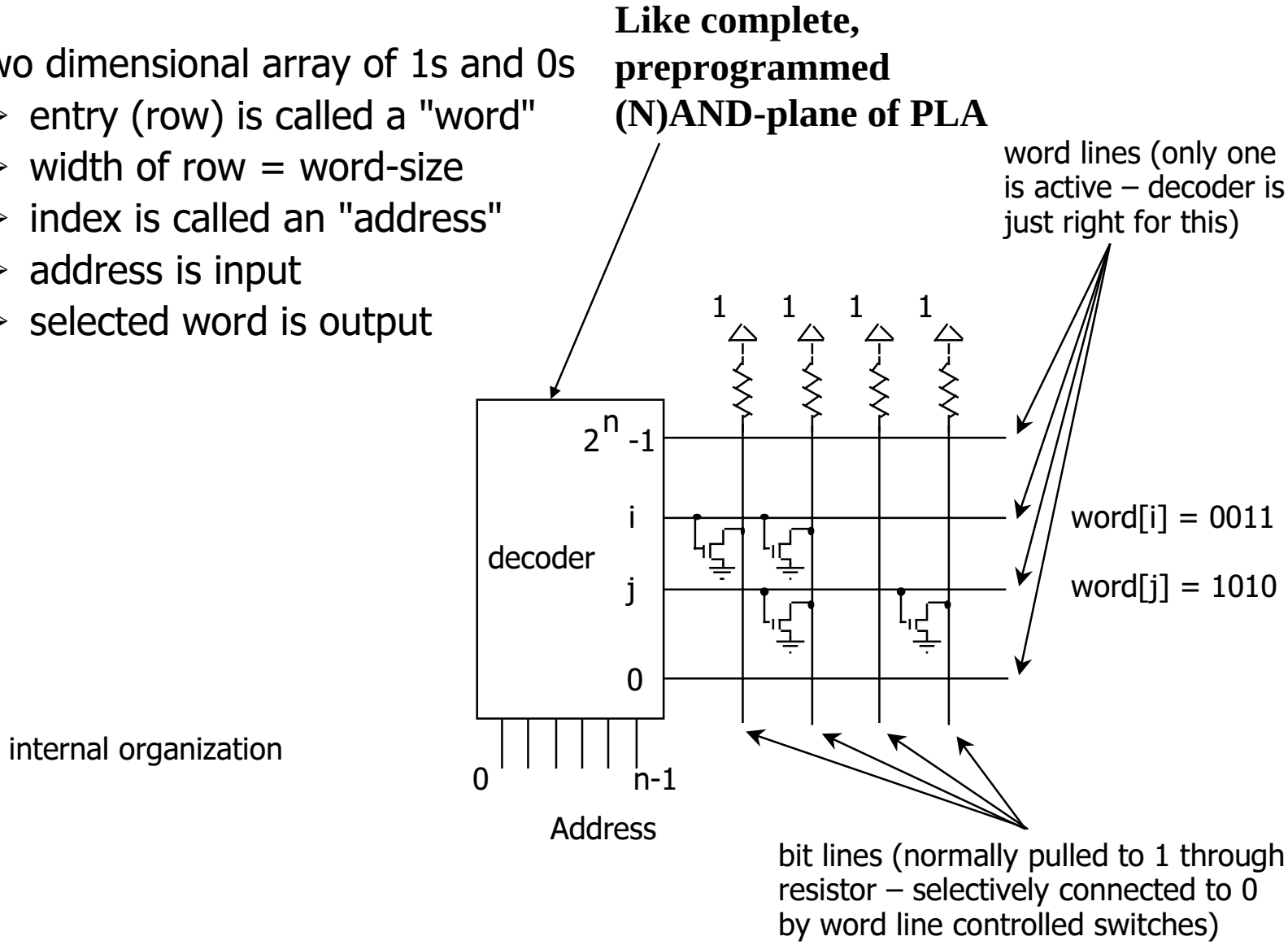
a given column of the OR array has access to only a subset of the possible product terms

If not using a row, then make zero



Read-only memories

- Two dimensional array of 1s and 0s
  - entry (row) is called a "word"
  - width of row = word-size
  - index is called an "address"
  - address is input
  - selected word is output



internal organization

ROMs and combinational logic

- Combinational logic implementation (two-level canonical form) using a ROM
- Put entire truth table into memory

F0 = A' B' C + A B' C' + A B' C

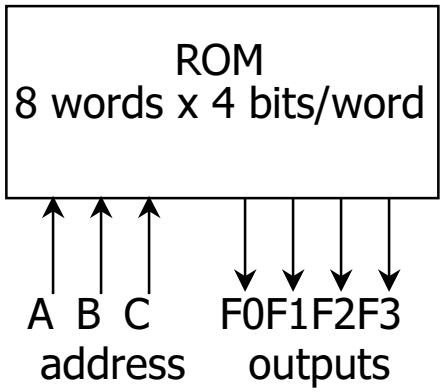
F1 = A' B' C + A' B C' + A B C

F2 = A' B' C' + A' B' C + A B' C'

F3 = A' B C + A B' C' + A B C'

A	B	C	F0	F1	F2	F3
0	0	0	0	0	1	0
0	0	1	1	1	1	0
0	1	0	0	1	0	0
0	1	1	0	0	0	1
1	0	0	1	0	1	1
1	0	1	1	0	0	0
1	1	0	0	0	0	1
1	1	1	0	1	0	0

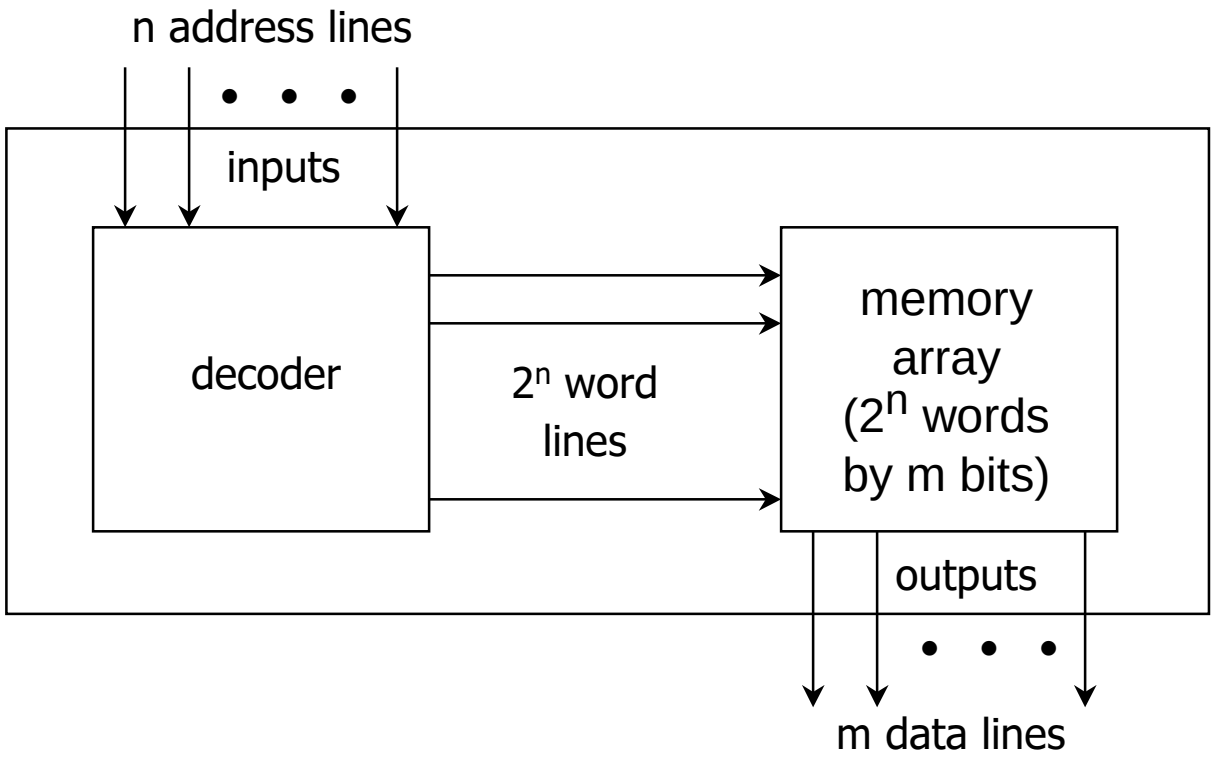
truth table



block diagram

ROM structure

- Similar to a PLA structure but with a fully decoded AND array
  - completely flexible OR array (unlike PAL)



ROM vs. PLA

- ROM approach advantageous when
  - design time is short (no need to minimize output functions)
  - most input combinations are needed (e.g., code converters)
  - little sharing of product terms among output functions
- ROM problems
  - size doubles for each additional input (32x4 for Calendar example)
  - can't exploit don't cares
- PLA approach advantageous when
  - design tools are available for multi-output minimization
  - there are relatively few unique minterm combinations
  - many minterms are shared among the output functions
  - Supports multilevel implementation using feedback
- PAL problems
  - constrained fan-ins on OR plane
  - Difficulty of common term re-use??

Regular logic structures for two-level logic

- ROM – full AND plane, general OR plane
  - cheap (high-volume component)
  - can implement any function of n inputs
  - medium speed
- PAL – programmable AND plane, fixed OR plane
  - intermediate cost
  - can implement functions limited by number of terms
  - high speed (only one programmable plane that is much smaller than ROM's decoder)
- PLA – programmable AND and OR planes
  - most expensive (most complex in design, need more sophisticated tools)
  - can implement any function up to a product term limit
  - slow (two programmable planes)

Regular logic structures for multi-level logic

- Difficult to devise a regular structure for arbitrary connections between a large set of different types of gates
  - efficiency/speed concerns for such a structure
  - in 467 you'll learn about field programmable gate arrays (FPGAs) that are just such programmable multi-level structures
    - programmable multiplexers for wiring
    - **lookup tables for logic functions (programming fills in the table)**
    - multi-purpose cells (utilization is the big issue)
- Use multiple levels of PALs/PLAs/ROMs
  - output intermediate result
  - make it an input to be used in further logic

Combinational logic implementation summary

- Multi-level logic
  - conversion to NAND-NAND and NOR-NOR networks
  - transition from simple gates to more complex gate building blocks
  - reduced gate count, fan-ins, potentially faster
  - more levels, harder to design
- Time response in combinational networks
  - gate delays and timing waveforms
  - hazards/glitches (what they are and why they happen)
- Regular logic
  - multiplexers/decoders
  - ROMs
  - PLAs/PALs
  - advantages/disadvantages of each