可编程逻辑阵列

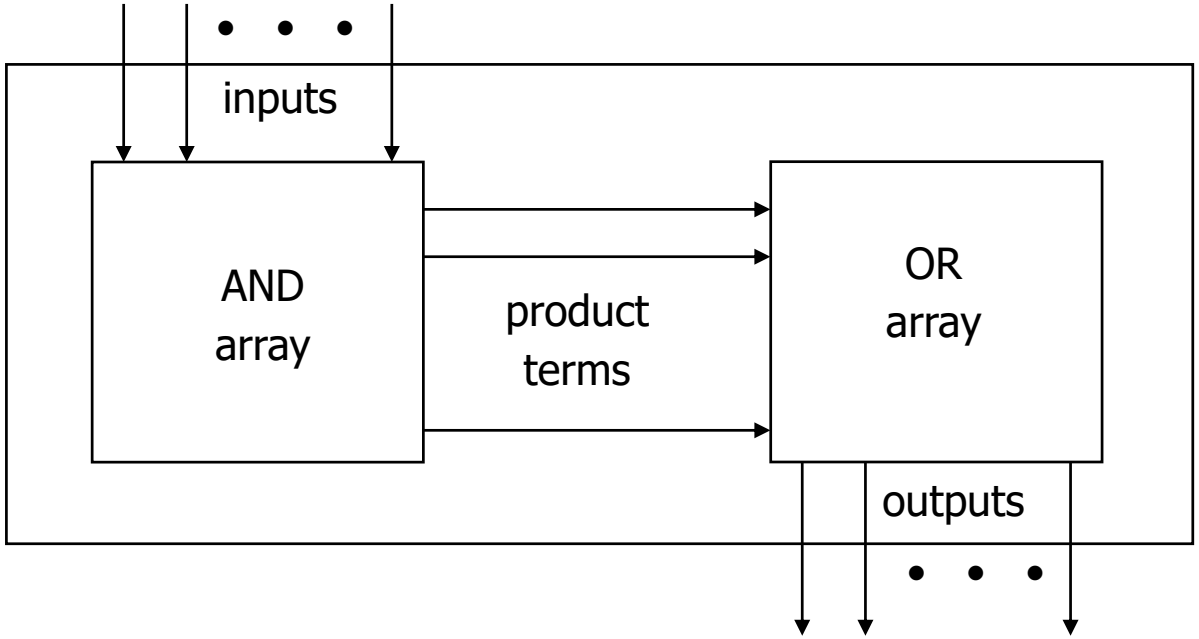
刘鹏

浙江大学信息与电子工程学院

liupeng@zju.edu.cn

可编程逻辑阵列（Programmable Logic Arrays，PLAs)

- Pre-fabricated building block of many AND/OR gates
  - Actually NOR or NAND
  - "Personalized" by making or breaking connections among gates
  - Programmable array block diagram for sum of products form



使能概念（Enabling Concept）

- Shared product terms among outputs

example:

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 F0	F1	F2	F3
AB	1	1	–	0	1	1	0
B'C	–	0	1	0	0	0	1
AC'	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:

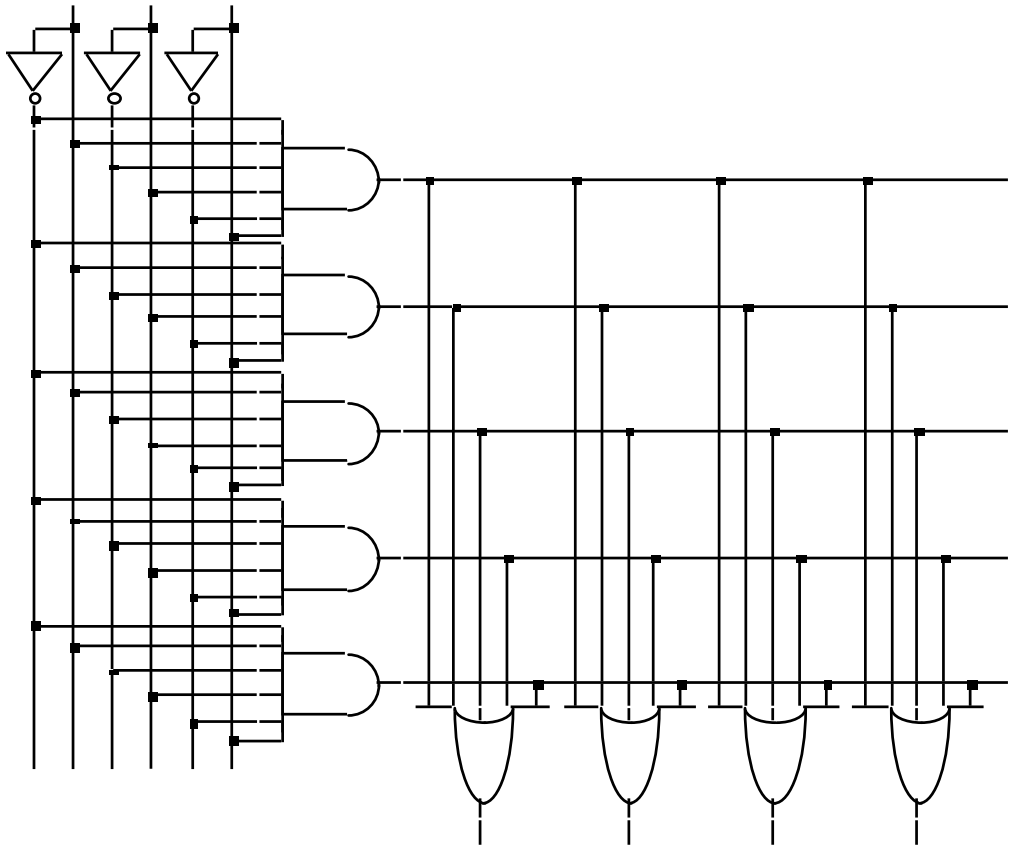
1 = uncomplemented in term
0 = complemented in term
– = does not participate

output side:

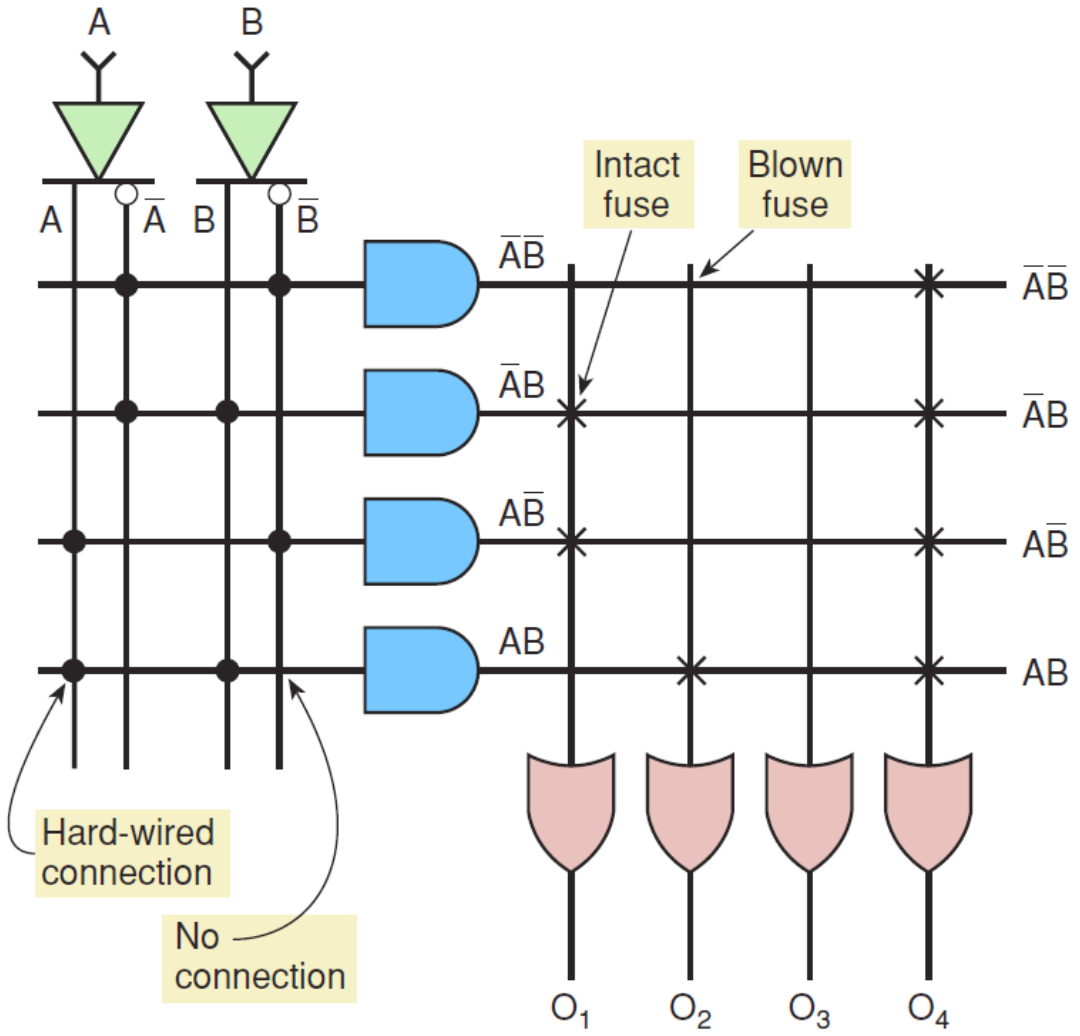
1 = term connected to output
0 = no connection to output

编程前

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

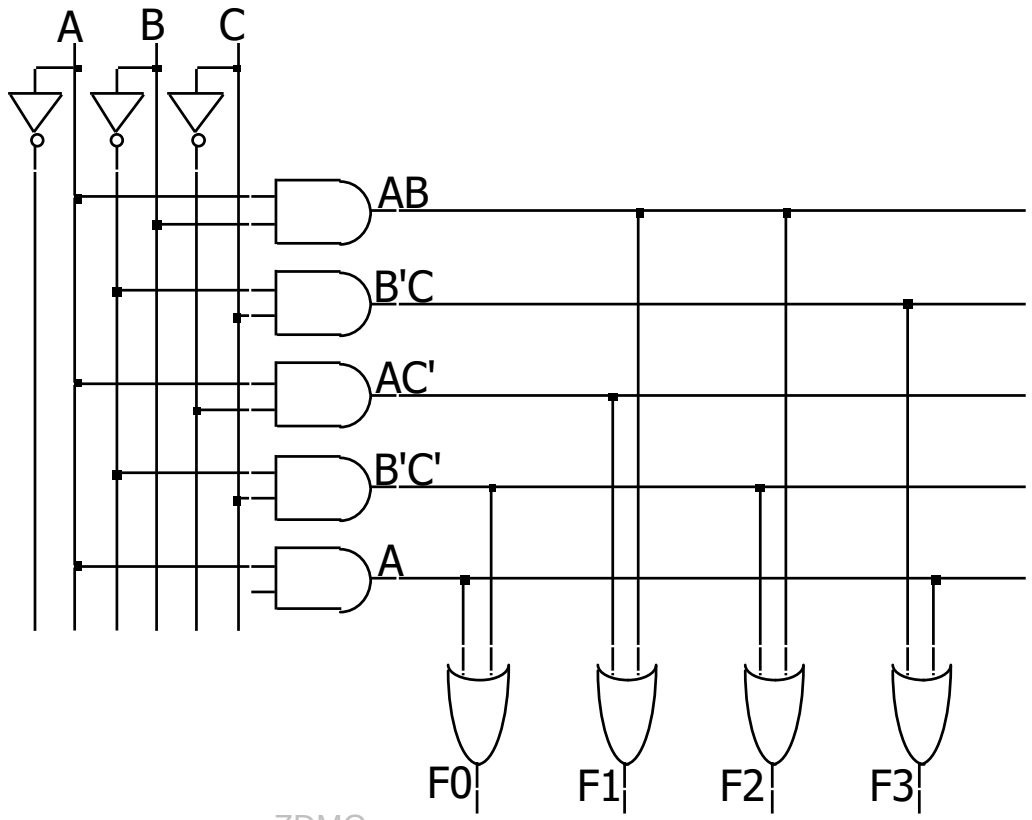


PLD符号


A
B
A
$\bar{A}$
B
$\bar{B}$
Intact fuse
Blown fuse
$\bar{A}\bar{B}$
$\bar{A}B$
$A\bar{B}$
AB
$\bar{A}\bar{B}$
$\bar{A}B$
$A\bar{B}$
AB
Hard-wired connection
No connection
O_1
O_2
O_3
O_4

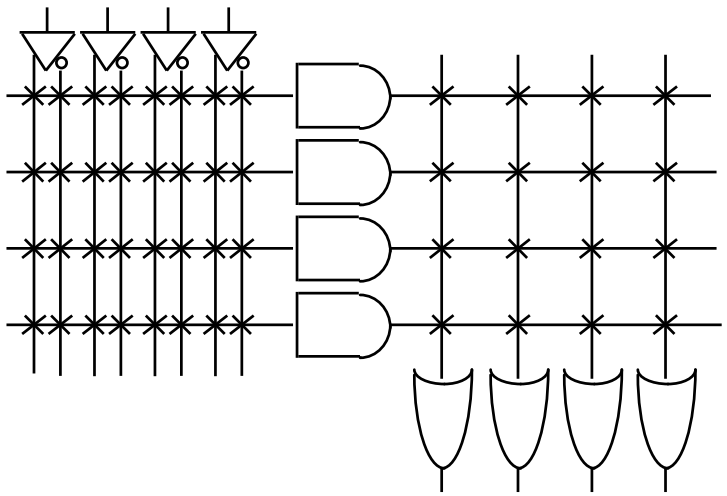
编程后

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

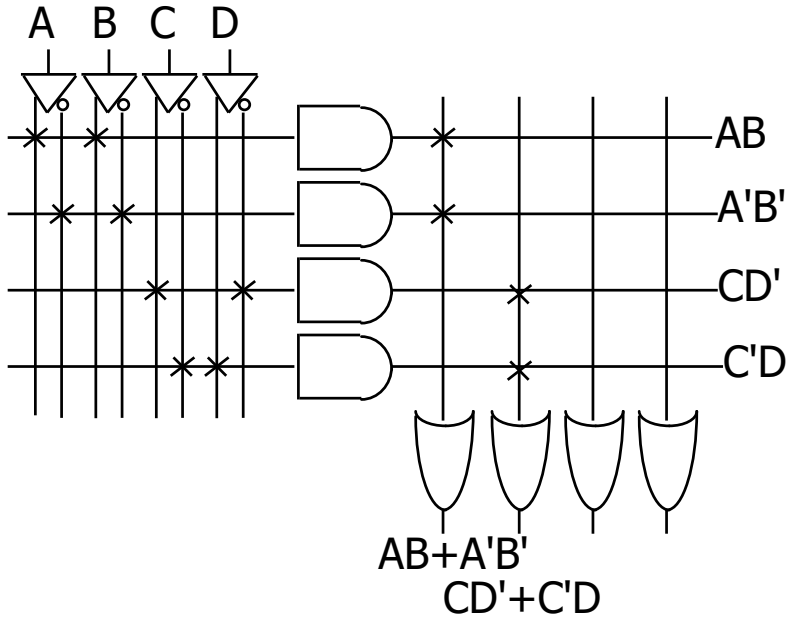


高扇入（Fan-in）结构的一种表示

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



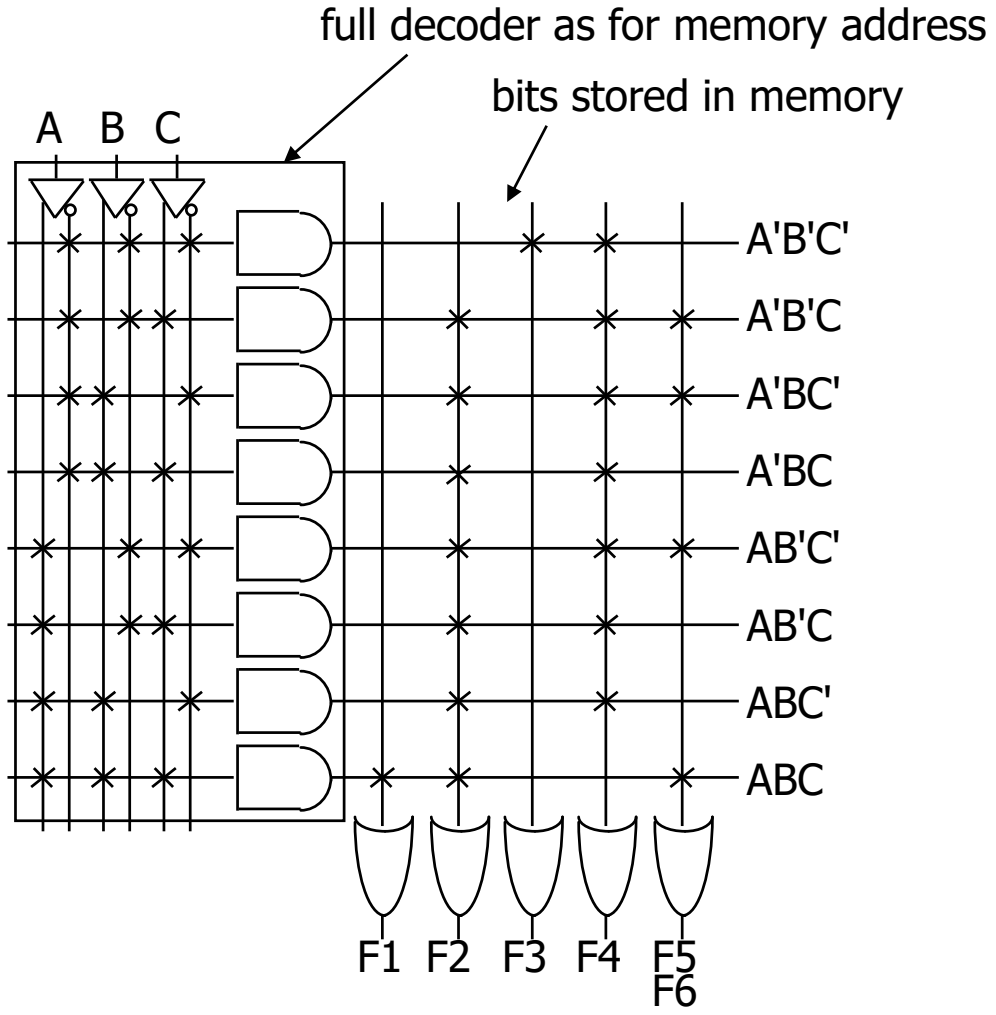
notation for implementing
F0 = A B + A' B'
F1 = C D' + C' D



可编程逻辑阵列例子

- 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
  - F6 = (A xnor B xnor C)'

A	B	C	F1	F2	F3	F4	F5	F6
0	0	0	0	0	1	1	0	0
0	0	1	0	1	0	1	1	1
0	1	0	0	1	0	1	1	1
0	1	1	0	1	0	1	0	0
1	0	0	0	1	0	1	1	1
1	0	1	0	1	0	1	0	0
1	1	0	0	1	0	1	0	0
1	1	1	1	1	0	0	1	1



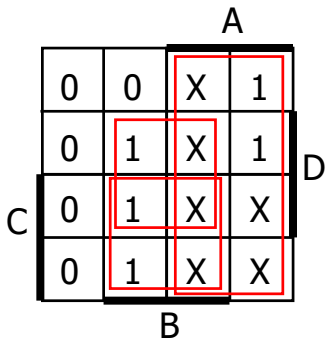
PLA 设计例子

- BCD to Gray code converter

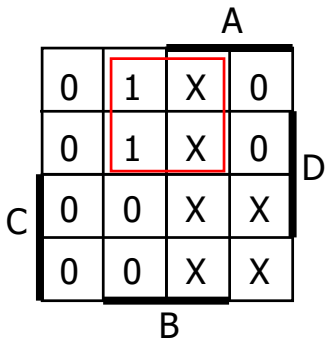
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	1	–	–	–	–	–
1	1	–	–	–	–	–	–

minimized functions:

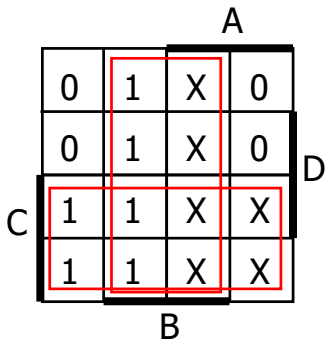
W = A + B D + B C
X = B C'
Y = B + C
Z = A'B'C'D + B C D + A D' + B' C D'



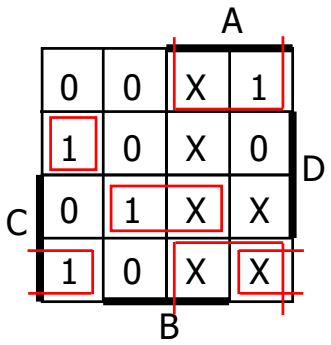
K-map for W



K-map for X



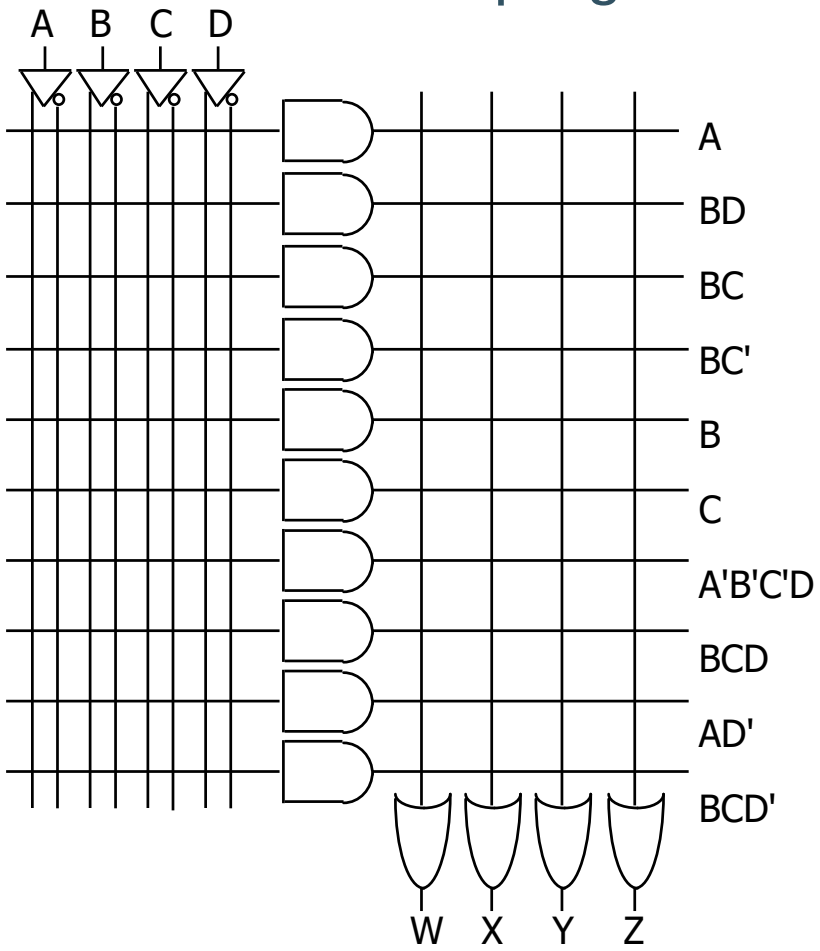
K-map for Y



K-map for Z

PLA 设计例子

- Code converter: programmed PLA



minimized functions:

W = A + B D + B C
X = B C'
Y = B + C
Z = A'B'C'D + B C D + A D' + B' C D'

not a particularly good candidate for PLA implementation since no terms are shared among outputs

however, much more compact and regular implementation when compared with discrete AND and OR gates

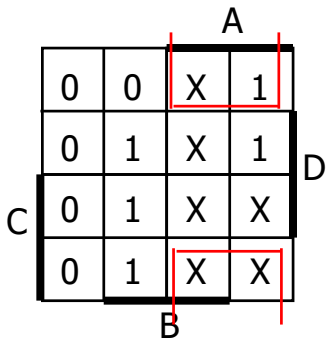
PLA设计例子

- BCD to Gray code converter

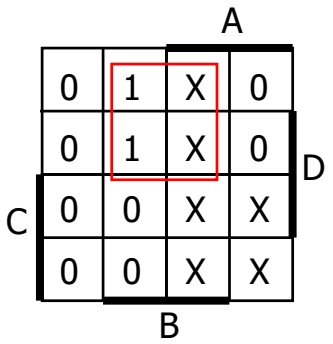
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	1	–	–	–	–	–
1	1	–	–	–	–	–	–

minimized functions:

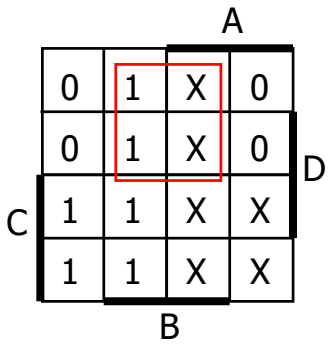
W =
X =
Y =
Z =



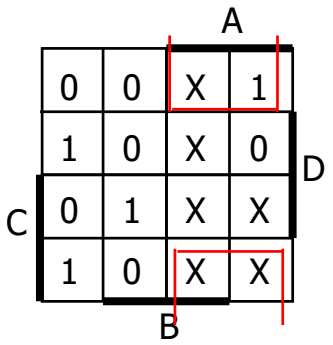
K-map for W



K-map for X



K-map for Y



K-map for Z

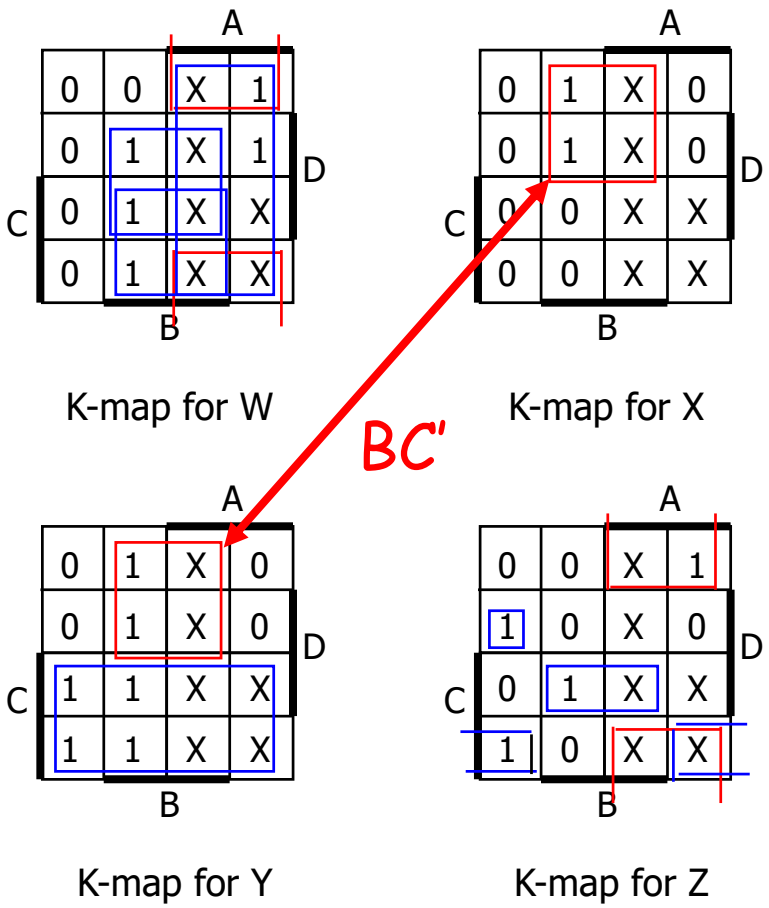
PLA 设计例子#1

- BCD to Gray code converter

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	–	–	–	–	–
1	1	–	–	–	–	–	–

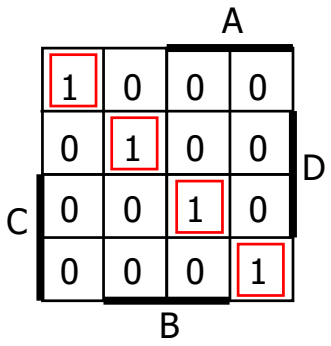
minimized functions:

W =
X =
Y =
Z =

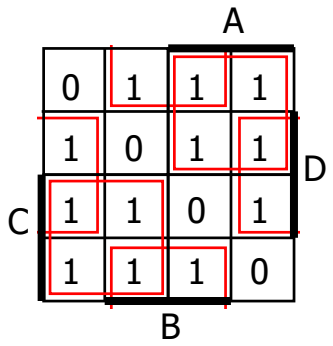


PLA 设计例子#2

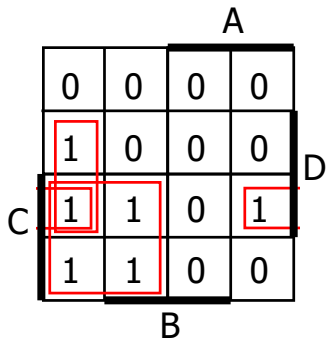
- Magnitude comparator



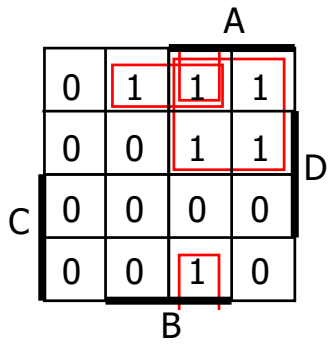
K-map for EQ



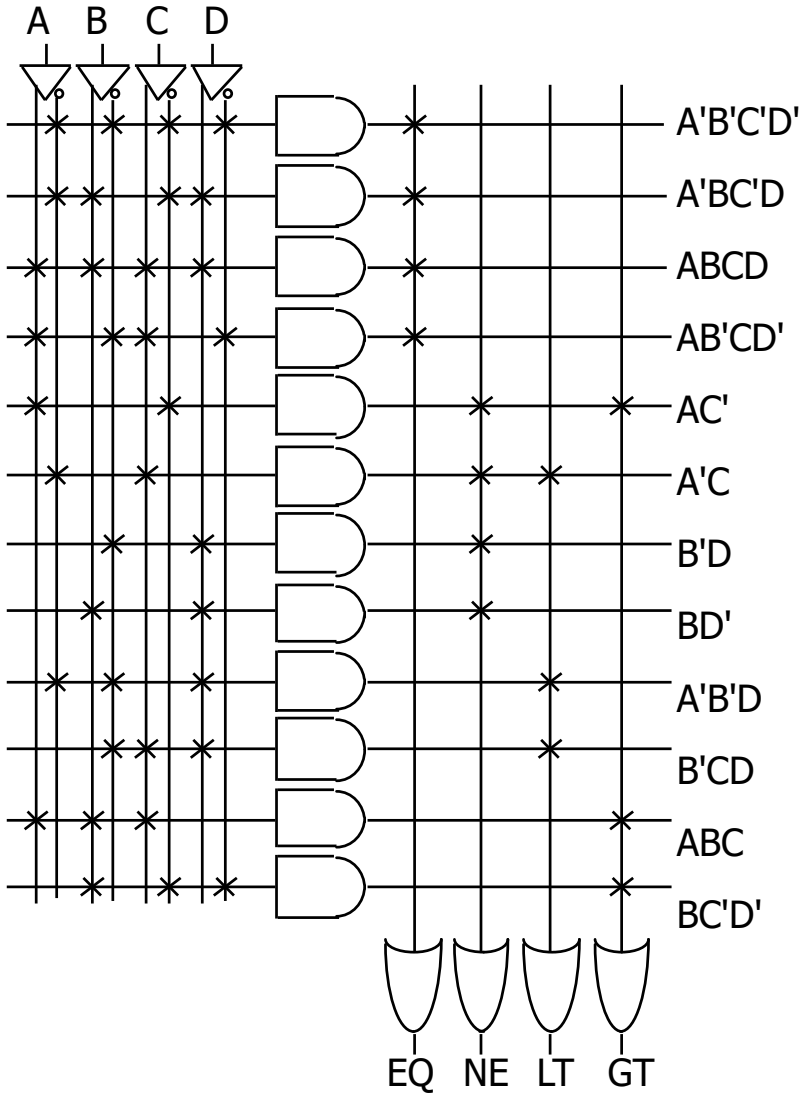
K-map for NE



K-map for LT

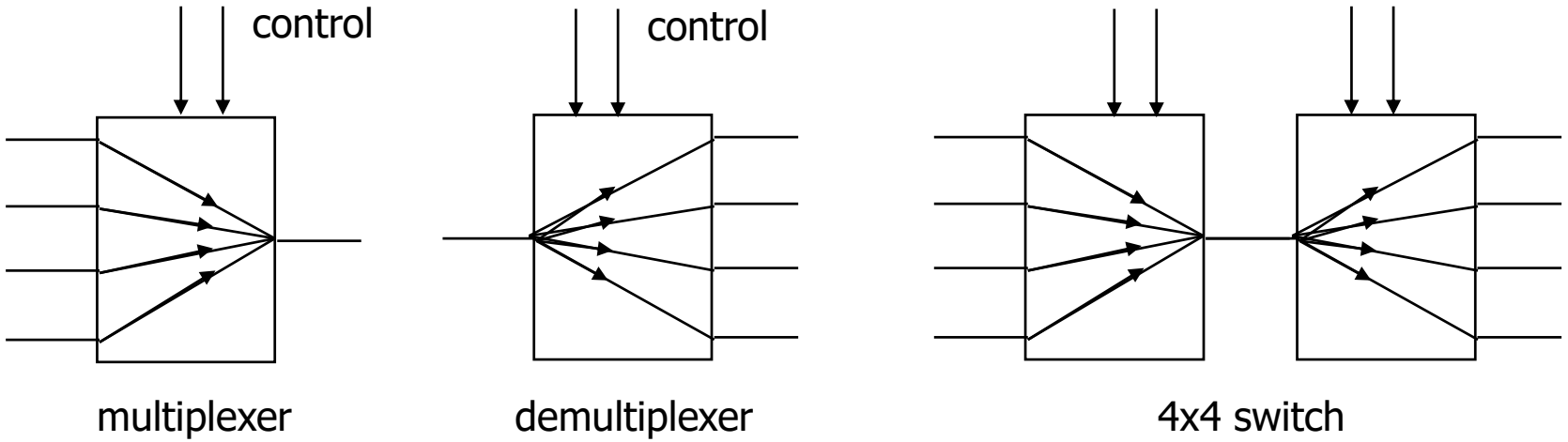


K-map for GT



多路选择器/解复器

- Direct point-to-point connections between gates
- *Multiplexer:* route one of many inputs to a single output
- *Demultiplexer:* route single input to one of many outputs



多路选择器

- Multiplexers/Selectors: general concept
  - 2^n data inputs, n control inputs (called "selects"), 1 output
  - Used to connect 2^n points to a single point
  - Control signal pattern forms binary index of input connected to output

$$Z = A' I_0 + A I_1$$

A	Z
0	I_0
1	I_1

functional form

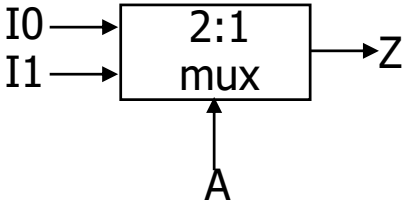
logical form

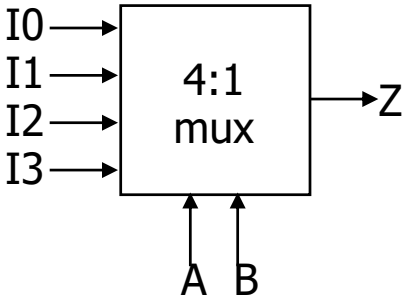
I_1	I_0	A	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

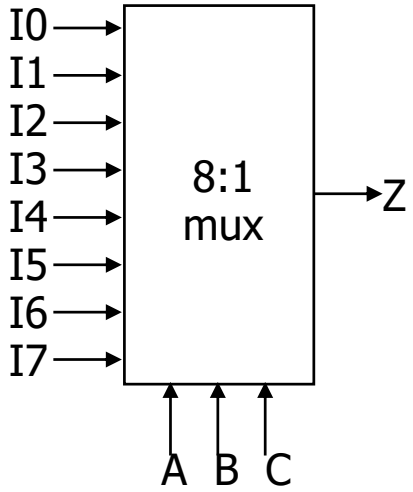
two alternative forms
for a 2:1 Mux truth table

多路选择器

- 2:1 mux: $Z = A' I0 + A I1$
- 4:1 mux: $Z = A' B' I0 + A' B I1 + A B' I2 + A B I3$
- 8:1 mux: $Z = A'B'C'I0 + A'B'CI1 + A'BC'I2 + A'BCI3 + AB'C'I4 + AB'CI5 + ABC'I6 + ABCI7$
- In general, $Z = \sum_{k=0}^{2^n - 1} (m_k I_k)$
  - in minterm shorthand form for a 2^n:1 Mux

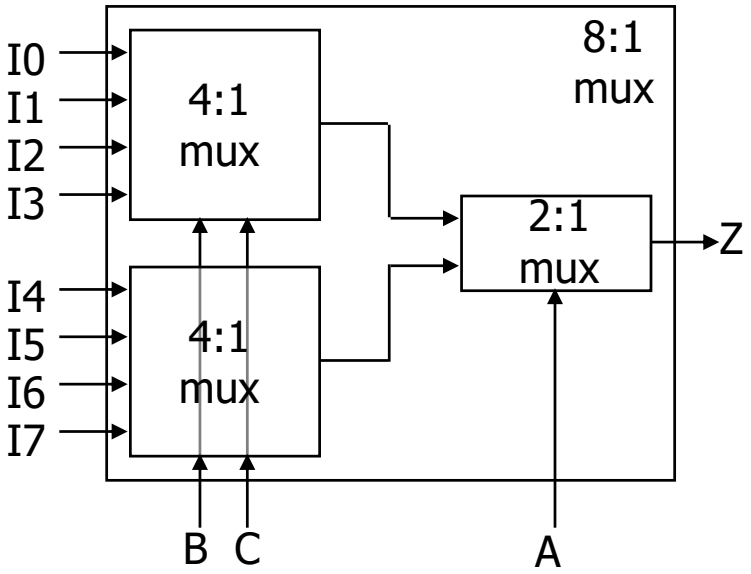






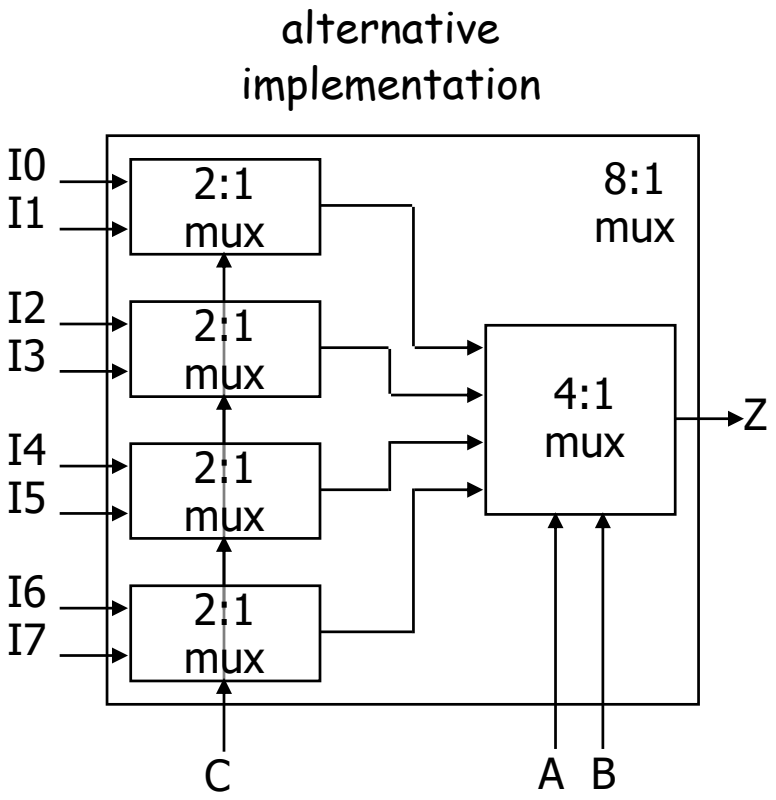
多路选择器级联

- Large multiplexers implemented by cascading smaller ones



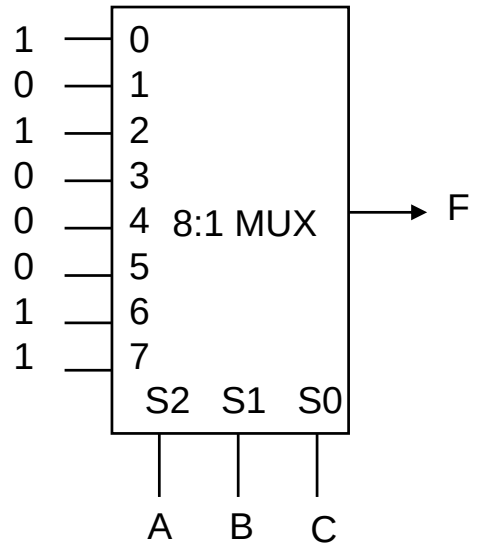
control signals B and C simultaneously choose one of I0, I1, I2, I3 and one of I4, I5, I6, I7

control signal A chooses which of the upper or lower mux's output to gate to Z



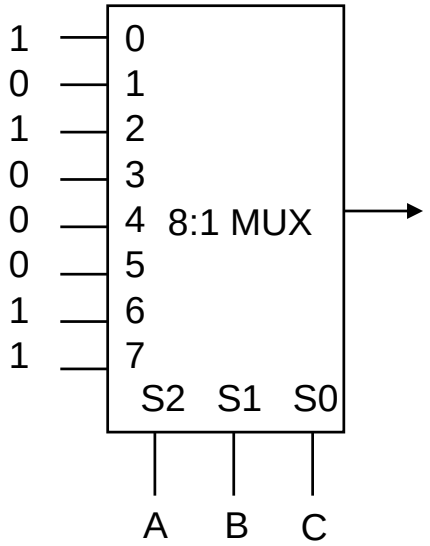
多路选择器作为查找表（Lookup Tables，LUTs）

- 2^n:1 multiplexer implements any function of n variables
  - With the variables used as control inputs and
  - Data inputs tied to 0 or 1
  - In essence, *a lookup table*
- Example:
  - F(A,B,C) = m0 + m2 + m6 + m7
    = A'B'C' + A'BC' + ABC' + ABC
    = A'B'(C') + A'B(C') + AB'(0) + AB(1)

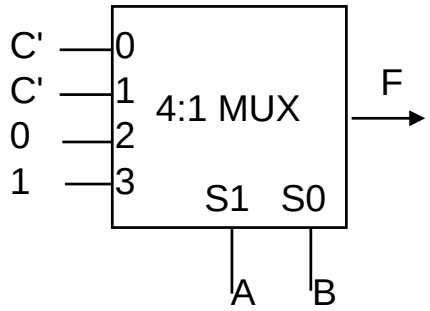


多路选择器作为查找表（Lookup Tables，LUTs）

- 2^{n-1}:1 mux can implement any function of n variables
  - With n-1 variables used as control inputs and
  - Data inputs tied to the last variable or its complement
- Example:
  - F(A,B,C) = m0 + m2 + m6 + m7
    = A'B'C' + A'BC' + ABC' + ABC
    = A'B'(C') + A'B(C') + AB'(0) + AB(1)

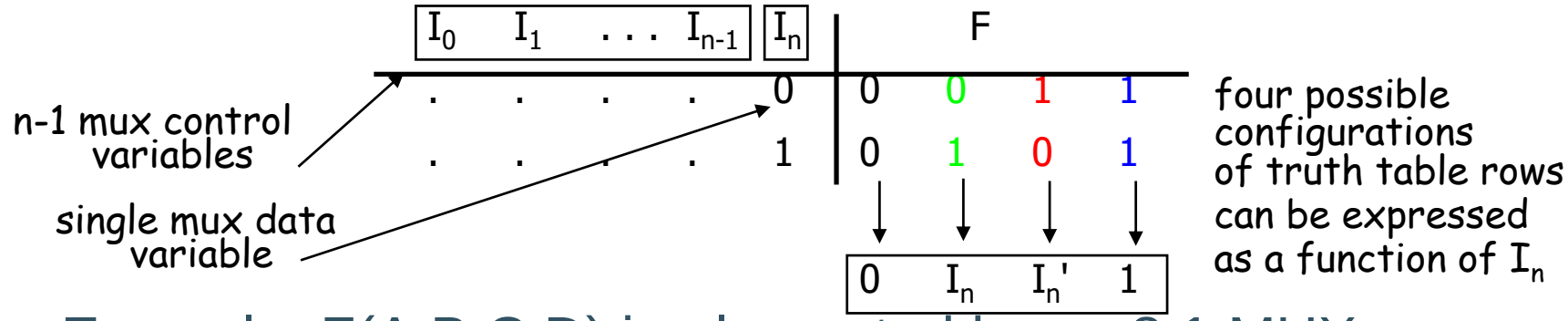


A	B	C	F	
0	0	0	1	C'
0	0	1	0	
0	1	0	1	C'
0	1	1	0	
1	0	0	0	0
1	0	1	0	
1	1	0	1	1
1	1	1	1	

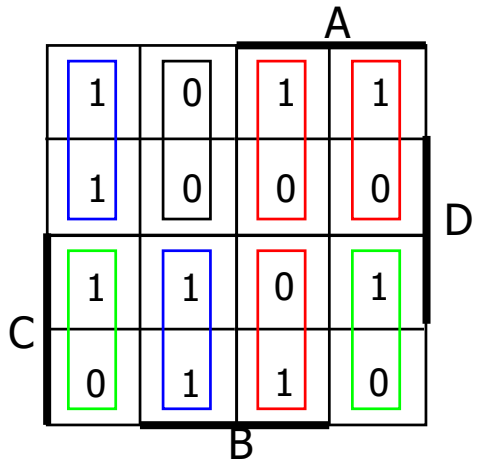


多路选择器作为查找表（Lookup Tables，LUTs）

- Generalization

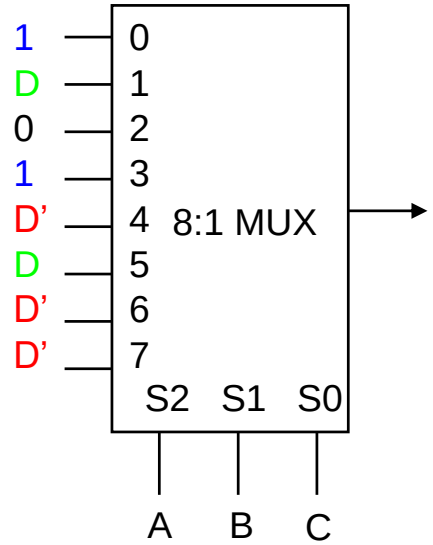


- Example: F(A,B,C,D) implemented by an 8:1 MUX



choose A,B,C as control variables

multiplexer implementation



解复器 / 译码器

- Decoders / demultiplexers: general concept
  - Single data input, n control inputs, 2^n outputs
  - Control inputs (called "selects" (S)) represent binary index of output to which the input is connected
  - Data input usually called "enable" (G)

1:2 Decoder:

O0 = G • S′
O1 = G • S

2:4 Decoder:

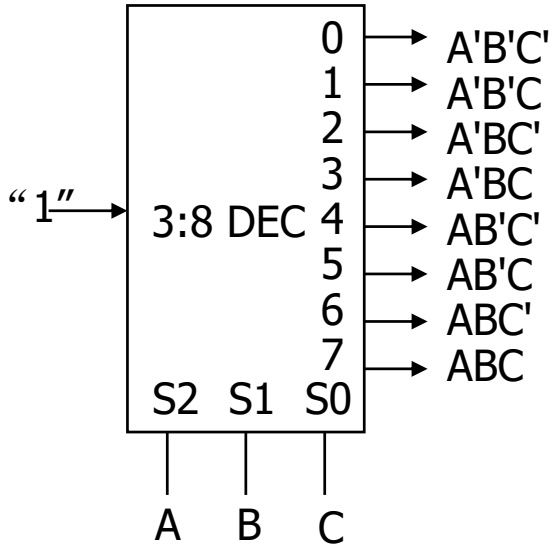
O0 = G • S1′ • S0′
O1 = G • S1′ • S0
O2 = G • S1 • S0′
O3 = G • S1 • S0

3:8 Decoder:

O0 = G • S2′ • S1′ • S0′
O1 = G • S2′ • S1′ • S0
O2 = G • S2′ • S1 • S0′
O3 = G • S2′ • S1 • S0
O4 = G • S2 • S1′ • S0′
O5 = G • S2 • S1′ • S0
O6 = G • S2 • S1 • S0′
O7 = G • S2 • S1 • S0

解复器作为通用逻辑

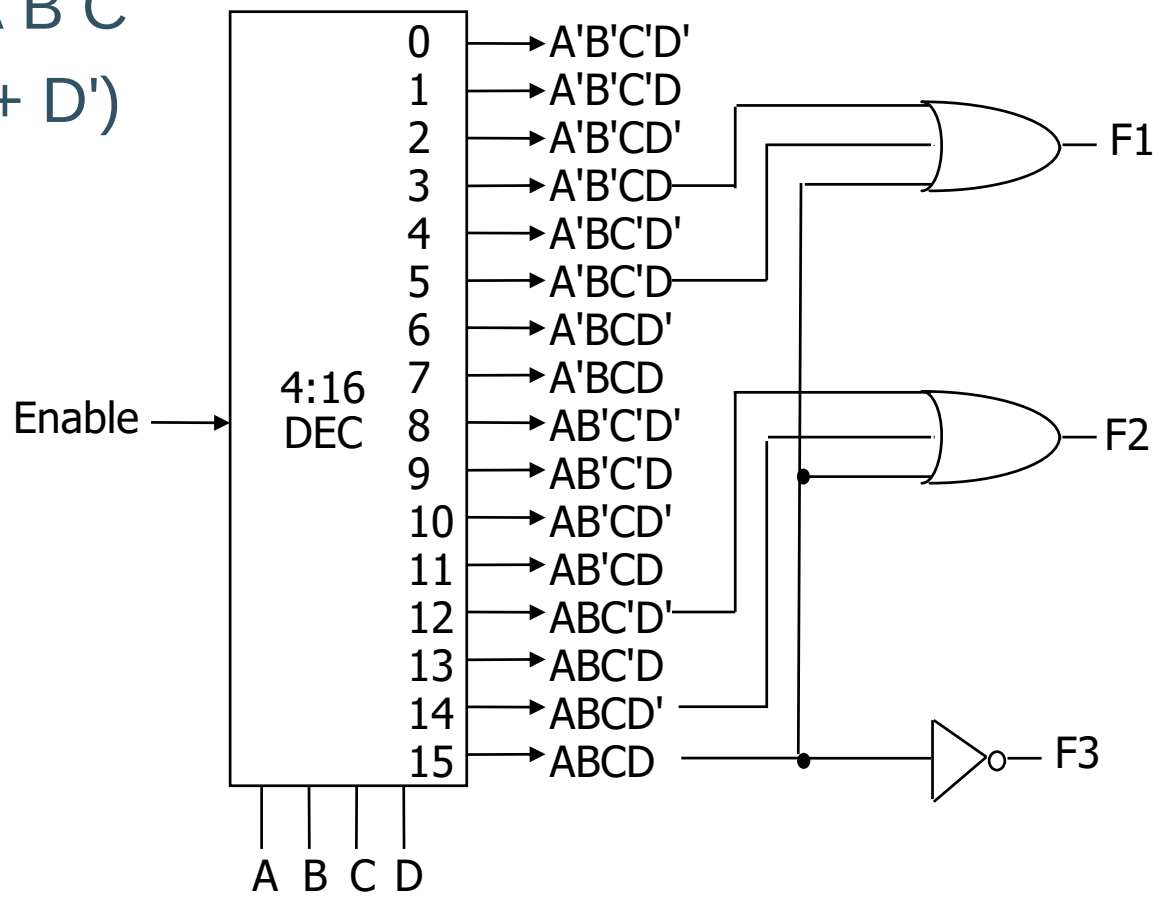
- n:2^n decoder implements any function of n variables
  - With the variables used as control inputs
  - Enable inputs tied to 1 and
  - Appropriate min-terms summed to form the function



demultiplexer generates appropriate
Min-term based on control signals
(it "decodes" control signals)

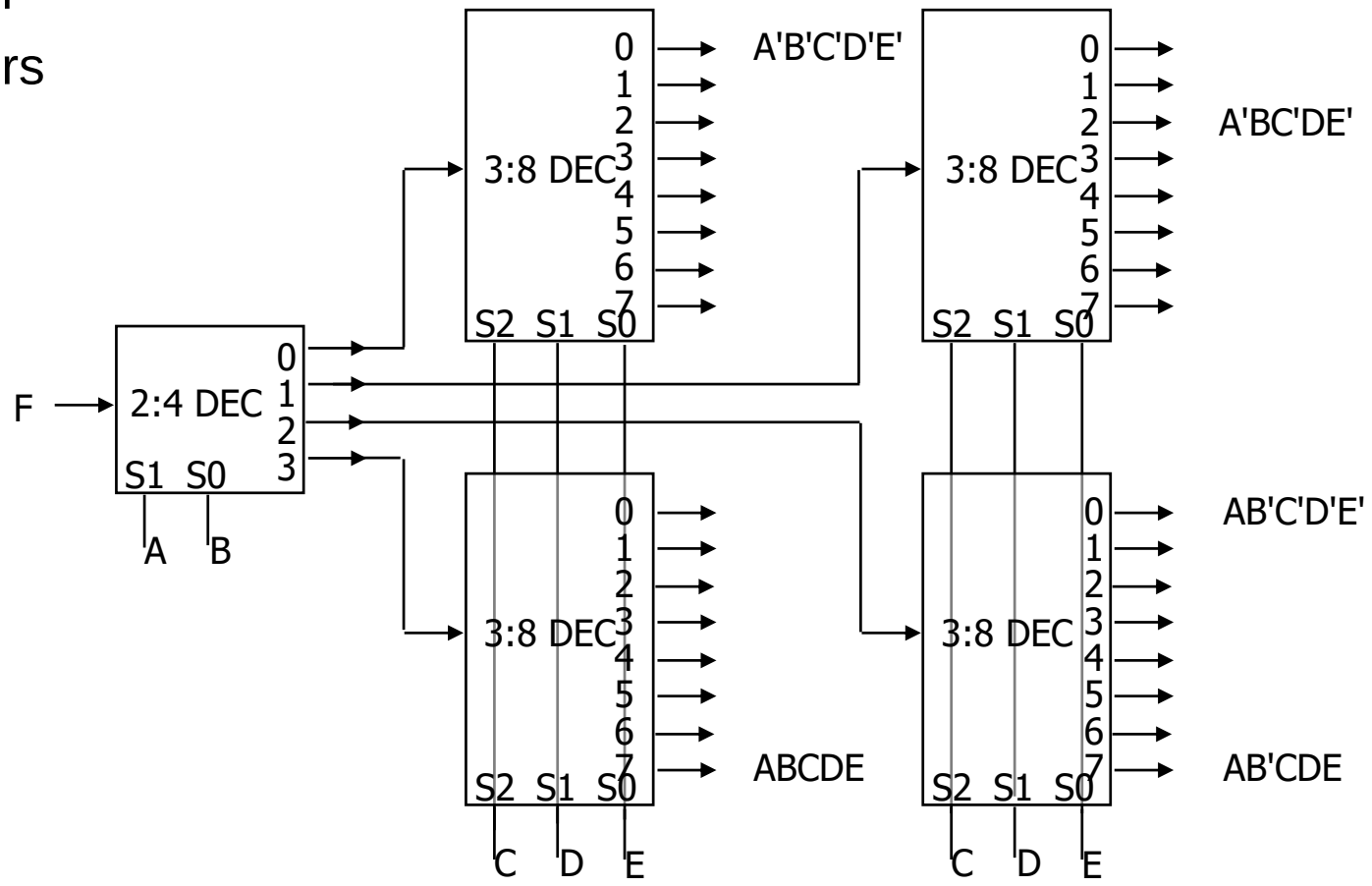
解复器作为通用逻辑

- F1 = A' B C' D + A' B' C D + A B C D
- F2 = A B C' D′ + A B C
- F3 = (A' + B' + C' + D')



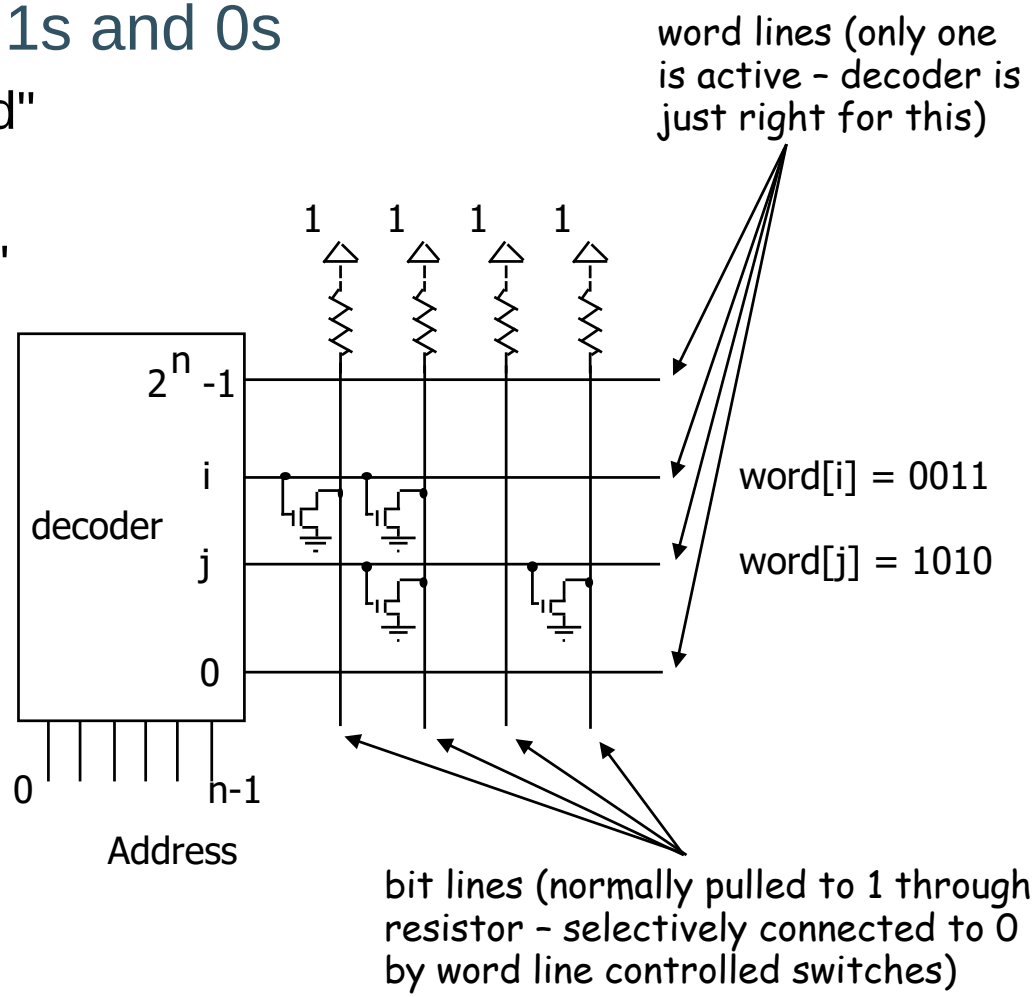
译码器级联

- 5:32 decoder
  - 1x2:4 decoder
  - 4x3:8 decoders



只读存储器（ROM）

- 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

ROM和组合逻辑

- Combinational logic implementation (two-level canonical form) using a ROM

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

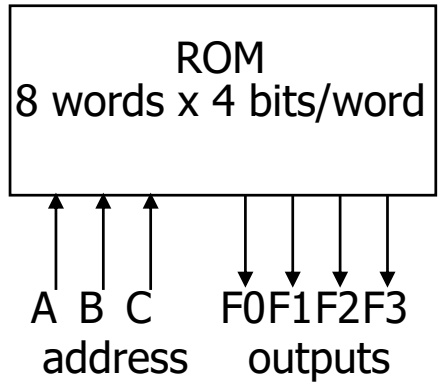
F1 = A' B' C + A' B C' + A B C

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

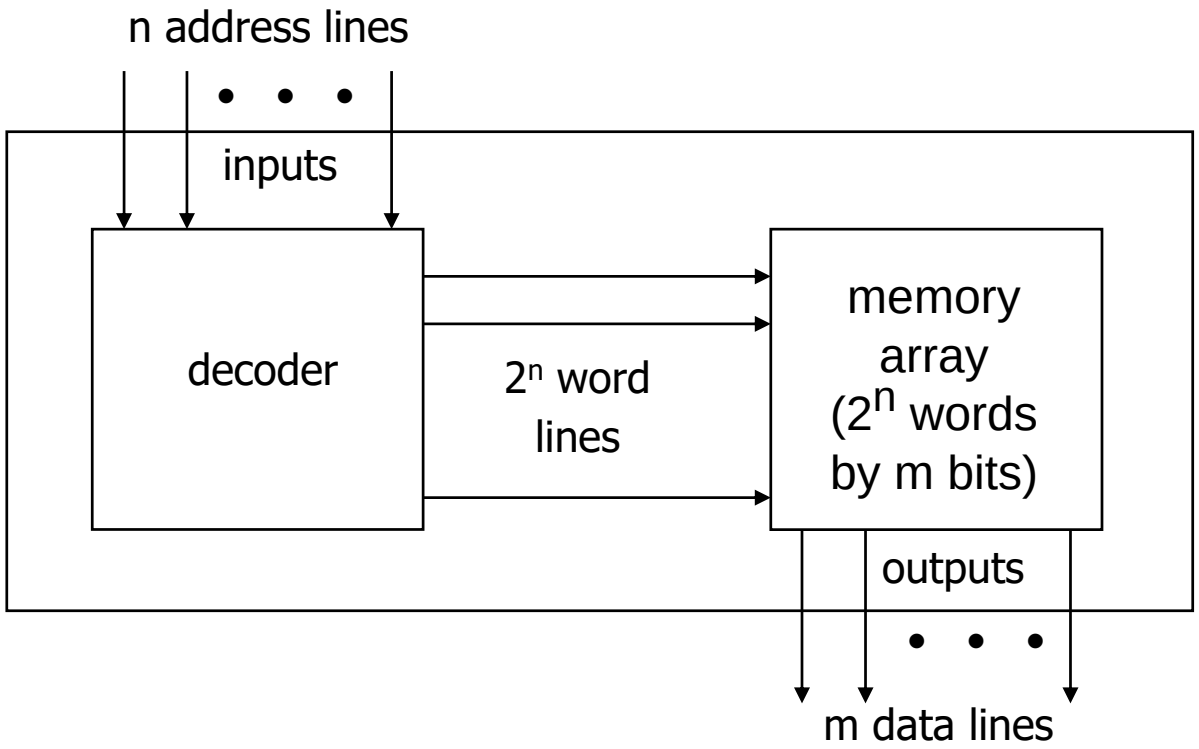
truth table



block diagram

ROM结构

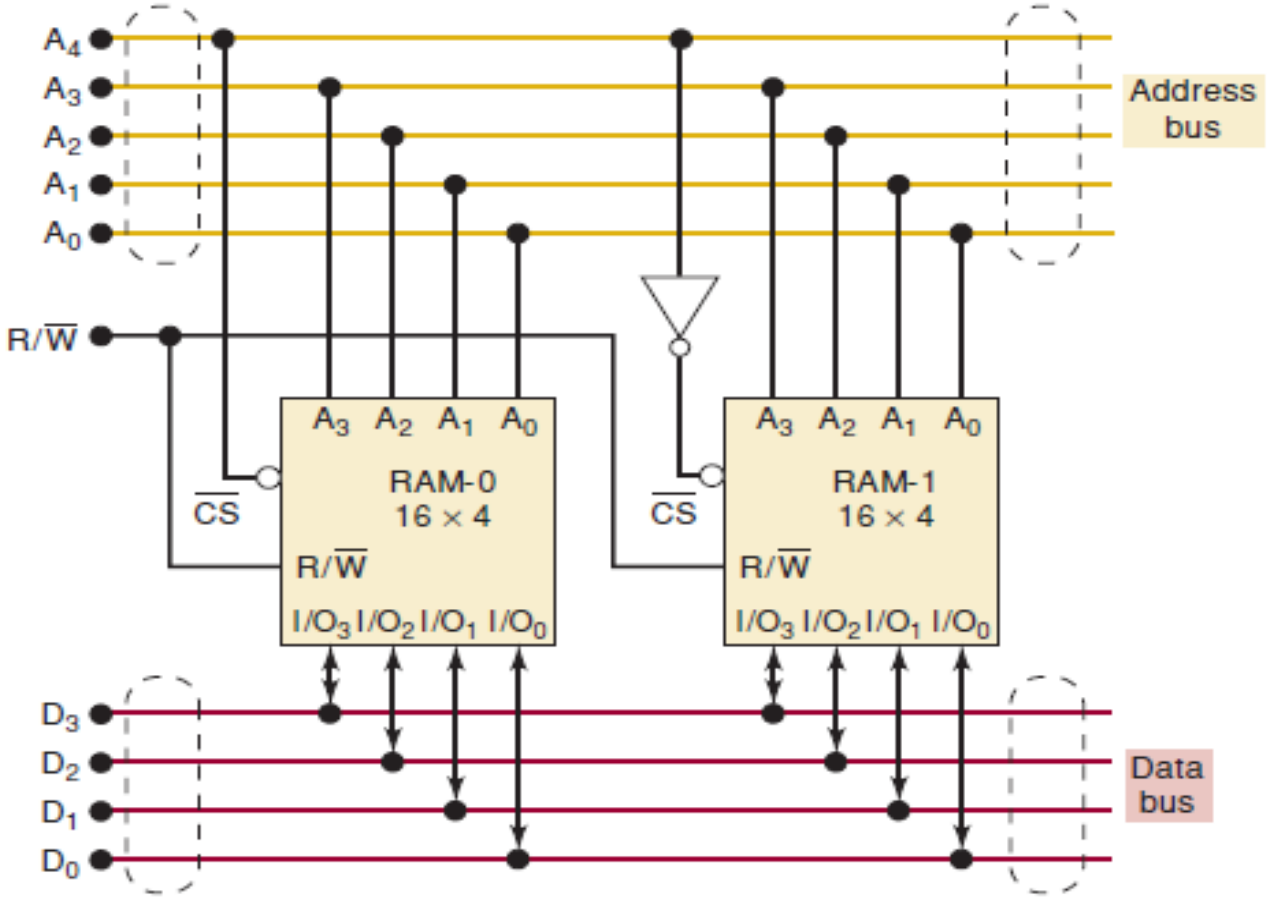
- Similar to a PLA structure but with a fully decoded AND array
  - Completely flexible OR array (unlike PAL)



ROM与 PLA对比

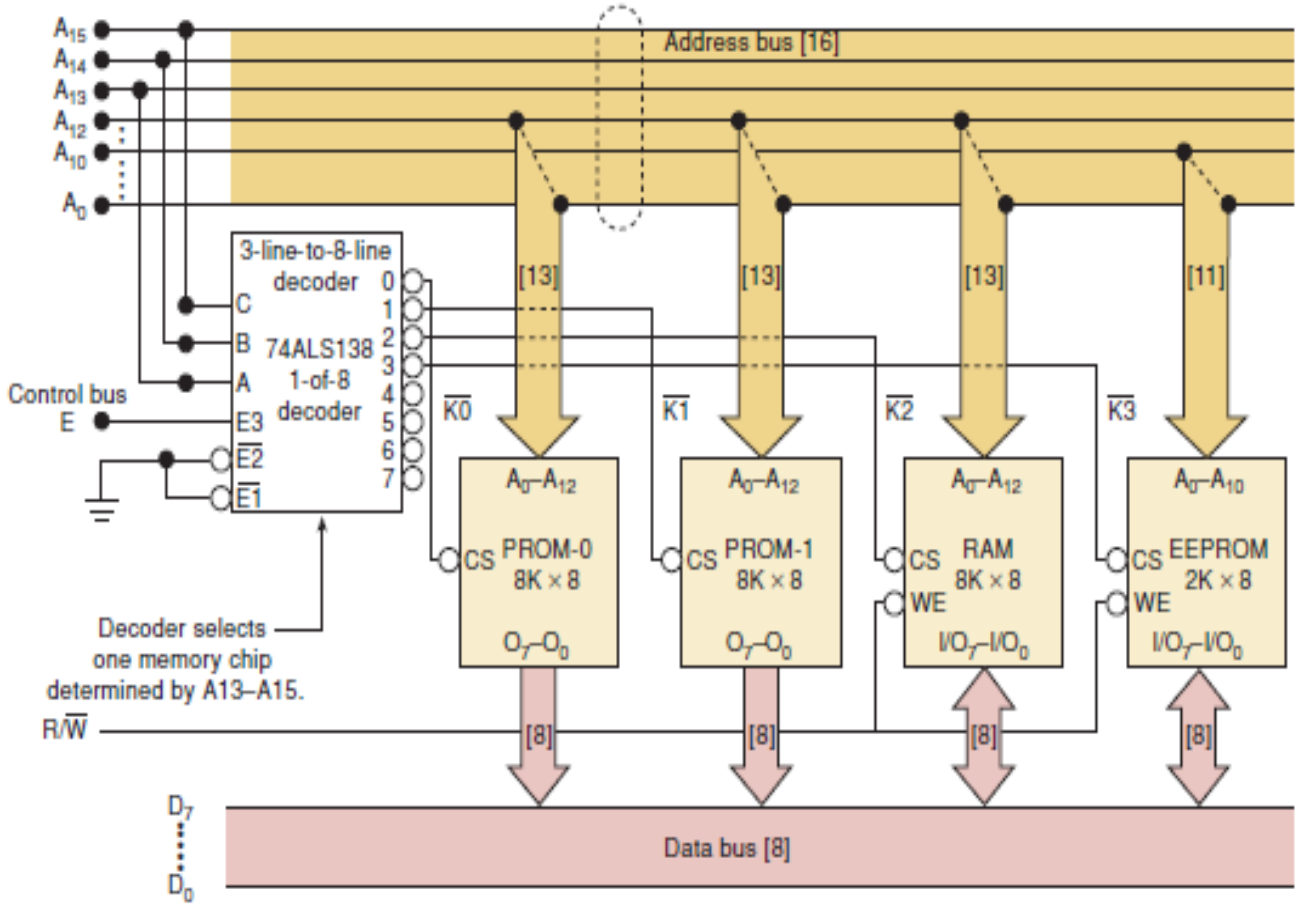
- ROM
  - Design time is short (no need to minimize output functions)
  - Most input combinations are needed (e.g., code converters)
  - Little sharing of product terms among output functions
  - Size doubles for each additional input
  - Can't exploit don't cares
  - Cheap (high-volume component)
  - Can implement any function of n inputs
  - Medium speed
- PLA
  - Design tools are available for multi-output minimization
  - There are relatively few unique min-term combinations
  - Many min-terms are shared among the output functions
  - Most complex in design, need more sophisticated tools
  - Can implement any function up to a product term limit
  - Slow (two programmable planes)

扩展字大小和存储容量



Address ranges:	00000 to 01111 – RAM-0 10000 to 11111 – RAM-1
Total	00000 to 11111 – (32 words)

地址译码的系统例子



Address ranges (hex)

0000 to 1FFF — PROM-0
2000 to 3FFF — PROM-1
4000 to 5FFF — RAM
6000 to 67FF — EEPROM