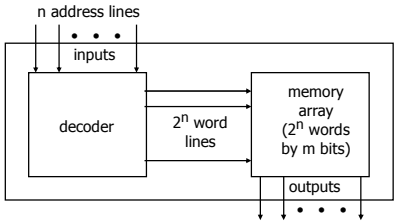
Lecture 10

- PLDs
  - ROMs
- Multilevel logic

Read-only memories (ROMs)

- Two dimensional array of stored 1s and 0s
  - Input is an address ⇒ ROM decodes all possible input addresses
  - Stored row entry is called a "word"
  - ROM output is the decoded word

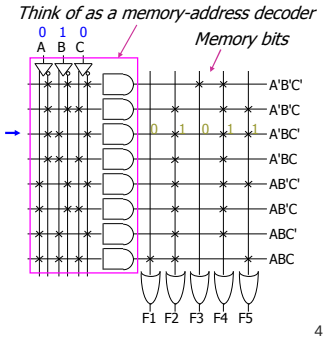
ROMs



Like this PLA example

F1 = ABC
F2 = A + B + C
F3 = A' B' C'
F4 = A' + B' + C'
F5 = A xor B xor C

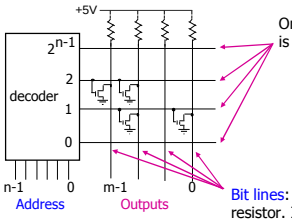
A	B	C	F1	F2	F3	F4	F5
0	0	0	0	0	1	1	0
0	0	1	0	1	0	1	1
0	1	0	0	1	0	1	1
0	1	1	0	1	0	1	0
1	0	0	0	1	0	1	1
1	0	1	0	1	0	1	0
1	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1



ROM details

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

ROM details



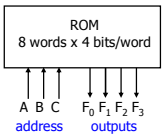
Only one word line is active at any time

Bit lines: Normally pulled high through resistor. If bit stores a zero, then transistor pulls low when row is selected.

Two-level logic using a ROM

- Use a ROM to directly store a truth table
  - No need to minimize logic

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



You specify whether to store 1 or 0 in each location in the ROM

ROMs versus PLAs/PALs

ROMs:

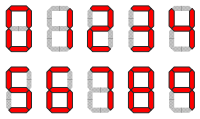
- Benefits
  - Quick to design, simple
- Limitations
  - Size doubles for each additional input
  - Can't exploit don't cares

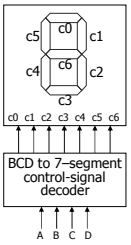
PLAs/PALs:

- Benefits
  - Logic minimization reduces size
- Limitations
  - PAL OR-plane has hard-wired fan-in

Example: BCD to 7-segments

- The problem
  - Input is a 4-bit BCD digit (A, B, C, D)
  - Need signals to drive a display (7 outputs C0 – C6)





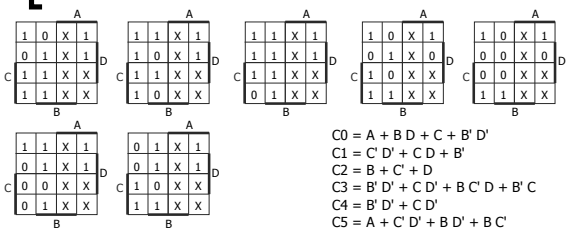
Formalize the problem

- Truth table
  - Many don't cares
- Choose implementation target
  - If ROM, we are done
  - Don't cares imply PAL/PLA may be good choice
- Implement design
  - Minimize the logic
  - Map into PAL/PLA

A	B	C	D	C0	C1	C2	C3	C4	C5	C6
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	X	X	X	X	X	X	X	X
1	1	X	X	X	X	X	X	X	X	X

Not all rows of the truth table are listed separately

SOP implementation



C0 = A + B D + C + B' D'
C1 = C' D' + C D + B'
C2 = B + C' + D
C3 = B' D' + C D' + B C' D + B' C
C4 = B' D' + C D'
C5 = A + C' D' + B D' + B C'
C6 = A + C D' + B C' + B' C

4 input, 7 output

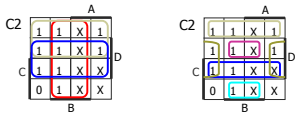
PLA: 15 AND gates

PAL: 4 product terms per output (28 AND gates)

- 15 unique product terms if we minimize individually

Better SOP for PLA

- Can do better than 15 product terms
  - Share terms among outputs ⇒ only 9 unique product terms
  - Each output not necessarily minimized
- For example:



Better SOP for PLA

- 15 unique product terms if minimized individually

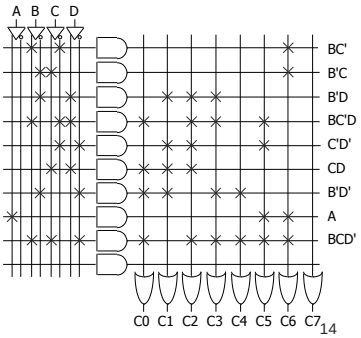
C0 = A + BD + C + B'D'
C1 = C'D' + CD + B'
C2 = B + C' + D
C3 = B'D' + CD' + BC'D + B'C
C4 = B'D' + CD'
C5 = A + C'D' + BD' + BC'
C6 = A + CD' + BC' + B'C

- 9 unique product terms if we try to share terms among outputs

C0 = BC'D + CD + B'D' + BCD' + A
C1 = B'D + C'D' + CD + B'D'
C2 = B'D + BC'D + C'D' + CD + BCD'
C3 = BC'D + B'D + B'D' + BCD'
C4 = B'D' + BCD'
C5 = BC'D + C'D' + A + BCD'
C6 = B'C + BC' + BCD' + A

PLA implementation

C0 = BC'D + CD + B'D' + BCD' + A
C1 = B'D + C'D' + CD + B'D'
C2 = B'D + BC'D + C'D' + CD + BCD'
C3 = BC'D + B'D + B'D' + BCD'
C4 = B'D' + BCD'
C5 = BC'D + C'D' + A + BCD'
C6 = B'C + BC' + BCD' + A



Multilevel logic

- Basic idea: Simplify logic using more than 2 gate levels
  - Time–space (speed versus gate count) tradeoff
    - Will talk about the speed issue with timing diagram
- Two-level logic *usually*
  - Has smaller delays (faster circuits)
  - more gates and more wires (more circuit area)
- Multilevel logic *usually*
  - Has fewer gates (smaller circuits)
  - more gate delays (slower circuits)

Example

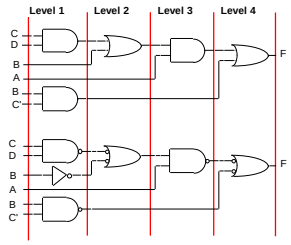
- SOP: X = ADF + AEF + BDF + BEF + CDF + CEF + G
  - X is minimized!
  - Six 3-input ANDs; one 7-input OR; 26 wires
- Multilevel: X = (A+B+C)(D+E)F + G
  - Factored form
  - One 3-input OR, two 2-input OR's, one 3-input AND; 11 wires

Multilevel NAND/NAND

F = A(B+CD) + BC'

original AND-OR network

introduce bubbles (conserve inversions)

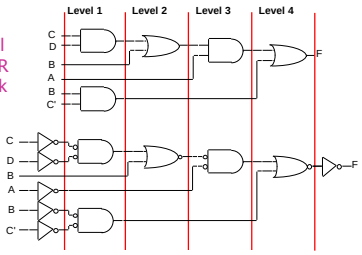


Multilevel NOR/NOR

F = A(B+CD) + BC'

original AND-OR network

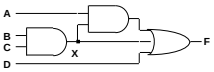
introduce bubbles (conserve inversions)



Generic multilevel conversion

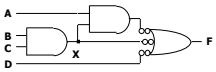
F = ABC + BC + D = AX + X + D

(a)



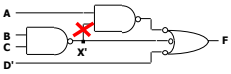
original circuit

(b)



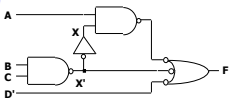
add double bubbles at inputs

(c)



distribute bubbles
some mismatches

(d)



insert inverters to fix mismatches

Issues with multilevel design

- No global definition of "optimal" multilevel circuit
  - Optimality depends on user-defined goals
- Synthesis requires CAD-tool help
  - No simple hand methods like K-maps
  - CAD tools manipulate Boolean expressions
  - Covered in more detail in CSE467

Multilevel logic summary

- Advantages over 2-level logic
  - Smaller circuits
  - Reduced fan-in
  - Less wires
- Disadvantages wrt 2-level logic
  - More difficult design
  - Less powerful optimizing tools