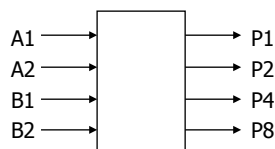


Working with Combinational Logic

- Simplification
 - two-level simplification
 - exploiting don't cares
 - algorithm for simplification
- Logic realization
 - two-level logic and canonical forms realized with NANDs and NORs
 - multi-level logic, converting between ANDs and ORs

Design example: 2x2-bit multiplier

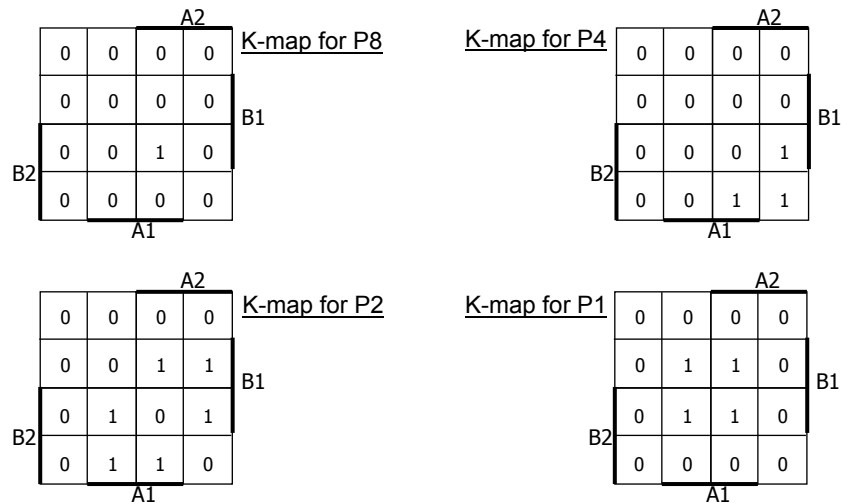


block diagram
and
truth table

A2	A1	B2	B1	P8	P4	P2	P1
0	0	0	0	0	0	0	0
		0	1	0	0	0	0
		1	0	0	0	0	0
		1	1	0	0	0	0
0	1	0	0	0	0	0	0
		0	1	0	0	0	1
		1	0	0	0	1	0
		1	1	0	0	1	1
1	0	0	0	0	0	0	0
		0	1	0	0	1	0
		1	0	0	1	0	0
		1	1	0	1	1	0
1	1	0	0	0	0	0	0
		0	1	0	0	1	1
		1	0	0	1	1	0
		1	1	1	0	0	1

4-variable K-map
for each of the 4
output functions

Design example: 2x2-bit multiplier (activity)



Spring 2010

CSE370 - VI - Logic Minimization

3

Definition of terms for two-level simplification

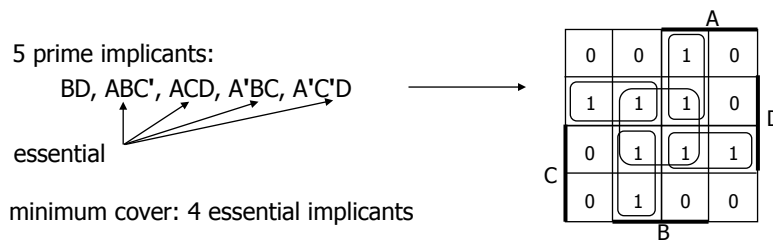
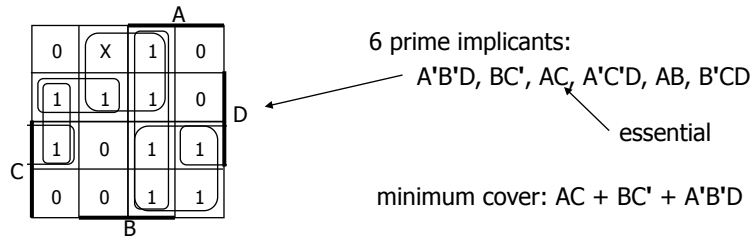
- **Implicant**
 - single element of ON-set or DC-set or any group of these elements that can be combined to form a subcube
- **Prime implicant**
 - implicant that can't be combined with another to form a larger subcube
- **Essential prime implicant**
 - prime implicant is essential if it alone covers an element of ON-set
 - will participate in ALL possible covers of the ON-set
 - DC-set used to form prime implicants but not to make implicant essential
- **Objective:**
 - grow implicant into prime implicants (minimize literals per term)
 - cover the ON-set with as few prime implicants as possible (minimize number of product terms)

Spring 2010

CSE370 - VI - Logic Minimization

4

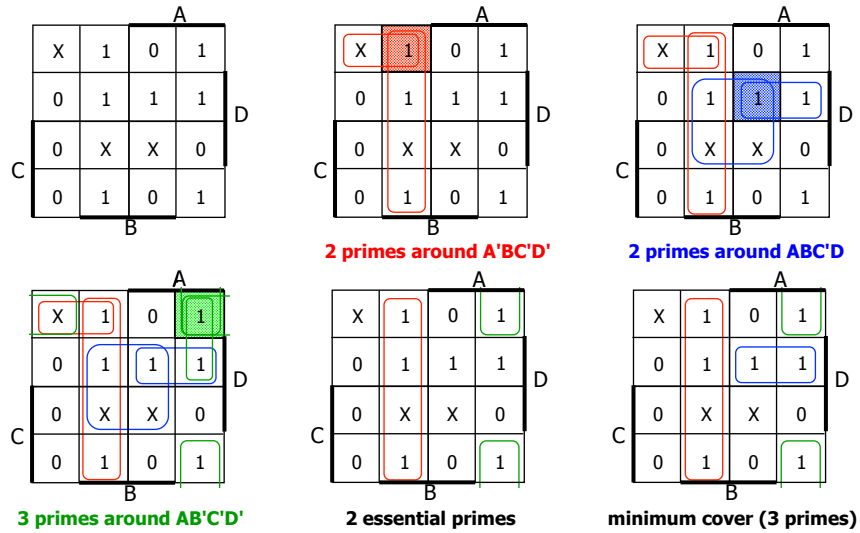
Examples to illustrate terms



Algorithm for two-level simplification

- Algorithm: minimum sum-of-products expression from a Karnaugh map
 - Step 1: choose an element of the ON-set
 - Step 2: find "maximal" groupings of 1s and Xs adjacent to that element
 - consider top/bottom row, left/right column, and corner adjacencies
 - this forms prime implicants (number of elements always a power of 2)
 - Repeat Steps 1 and 2 to find all prime implicants
 - Step 3: revisit the 1s in the K-map
 - if covered by single prime implicant, it is essential, and participates in final cover
 - 1s covered by essential prime implicant do not need to be revisited
 - Step 4: if there remain 1s not covered by essential prime implicants
 - select the smallest number of prime implicants that cover the remaining 1s

Algorithm for two-level simplification (example)



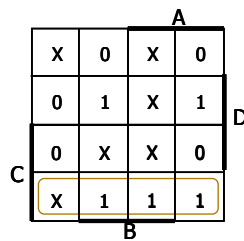
Spring 2010

CSE370 - VI - Logic Minimization

7

Activity

- List all prime implicants for the following K-map:



- Which are essential prime implicants?
- What is the minimum cover?

Spring 2010

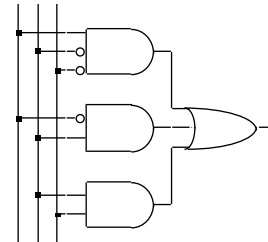
CSE370 - VI - Logic Minimization

8

Implementations of two-level logic

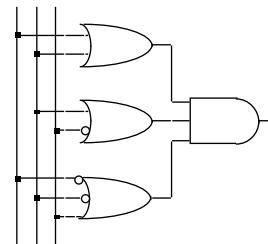
- Sum-of-products

- AND gates to form product terms (minterms)
- OR gate to form sum



- Product-of-sums

- OR gates to form sum terms (maxterms)
- AND gates to form product



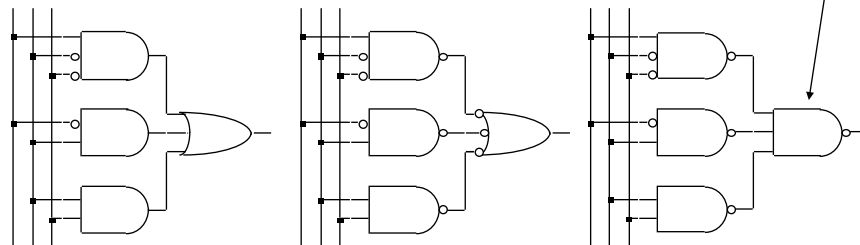
Two-level logic using NAND gates (cont'd)

- OR gate with inverted inputs is a NAND gate

- de Morgan's: $A' + B' = (A \cdot B)'$

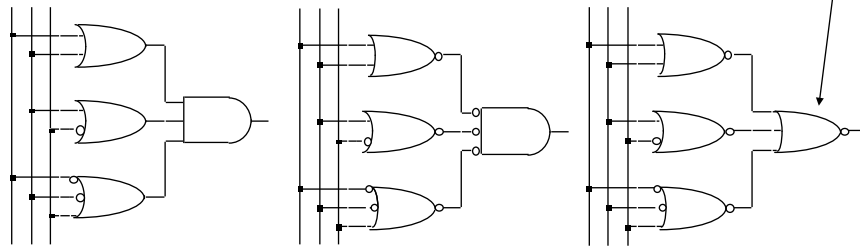
- Two-level NAND-NAND network

- inverted inputs are not counted
- in a typical circuit, inversion is done once and signal distributed



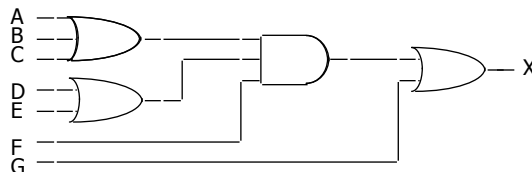
Two-level logic using NOR gates (cont'd)

- AND gate with inverted inputs is a NOR gate
 - de Morgan's: $A' \cdot B' = (A + B)'$
- Two-level NOR-NOR network
 - inverted inputs are not counted
 - in a typical circuit, inversion is done once and signal distributed



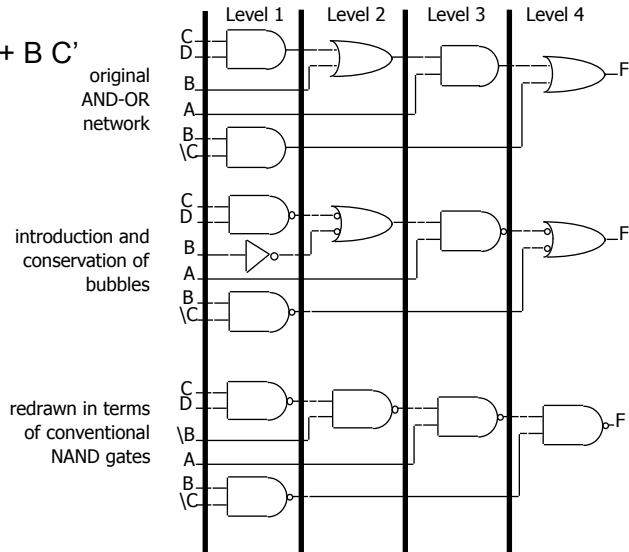
Multi-level logic

- $x = ADF + AEF + BDF + BEF + CDF + CEF + G$
 - reduced sum-of-products form – already simplified
 - 6 x 3-input AND gates + 1 x 7-input OR gate (that may not even exist!)
 - 25 wires (19 literals plus 6 internal wires)
- $x = (A + B + C)(D + E)F + G$
 - factored form – not written as two-level S-o-P
 - 1 x 3-input OR gate, 2 x 2-input OR gates, 1 x 3-input AND gate
 - 10 wires (7 literals plus 3 internal wires)



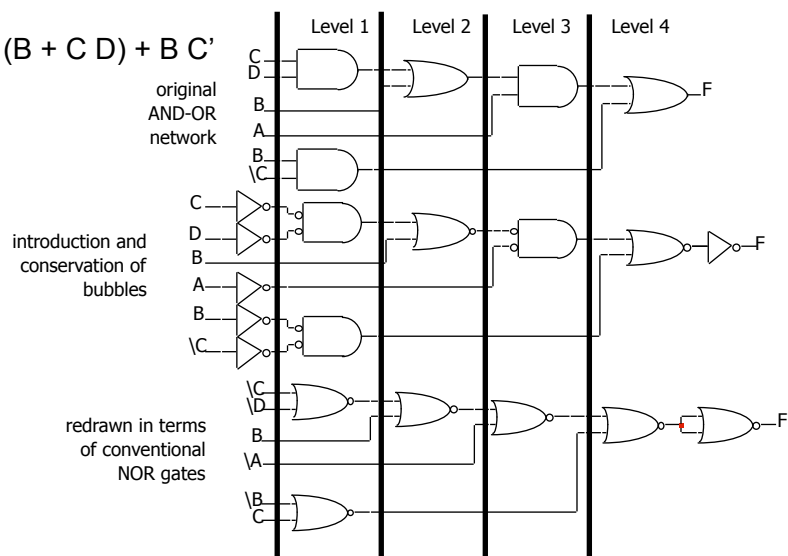
Conversion of multi-level logic to NAND gates

■ $F = A(B + CD) + BC'$



Conversion of multi-level logic to NORs

■ $F = A(B + CD) + BC'$



Summary for multi-level logic

- Advantages
 - circuits may be smaller
 - gates have smaller fan-in
 - circuits may be faster
- Disadvantages
 - more difficult to design
 - tools for optimization are not as good as for two-level
 - analysis is more complex