What is Good Logic Design?

Circuit Simplification - Part I

Learning Objectives

After this lecture, you should be able to…….

- **Explain the need for logic reduction**
- **Name the 3 methods of logic reduction**
- **Reduce logic expressions and circuits using**
  - **Boolean algebra theorems**
  - **Karnaugh-Maps**
- **Characterize propagation delays for a gate**
- **Use OR gate, MUX decomposition to simplify logic circuits**

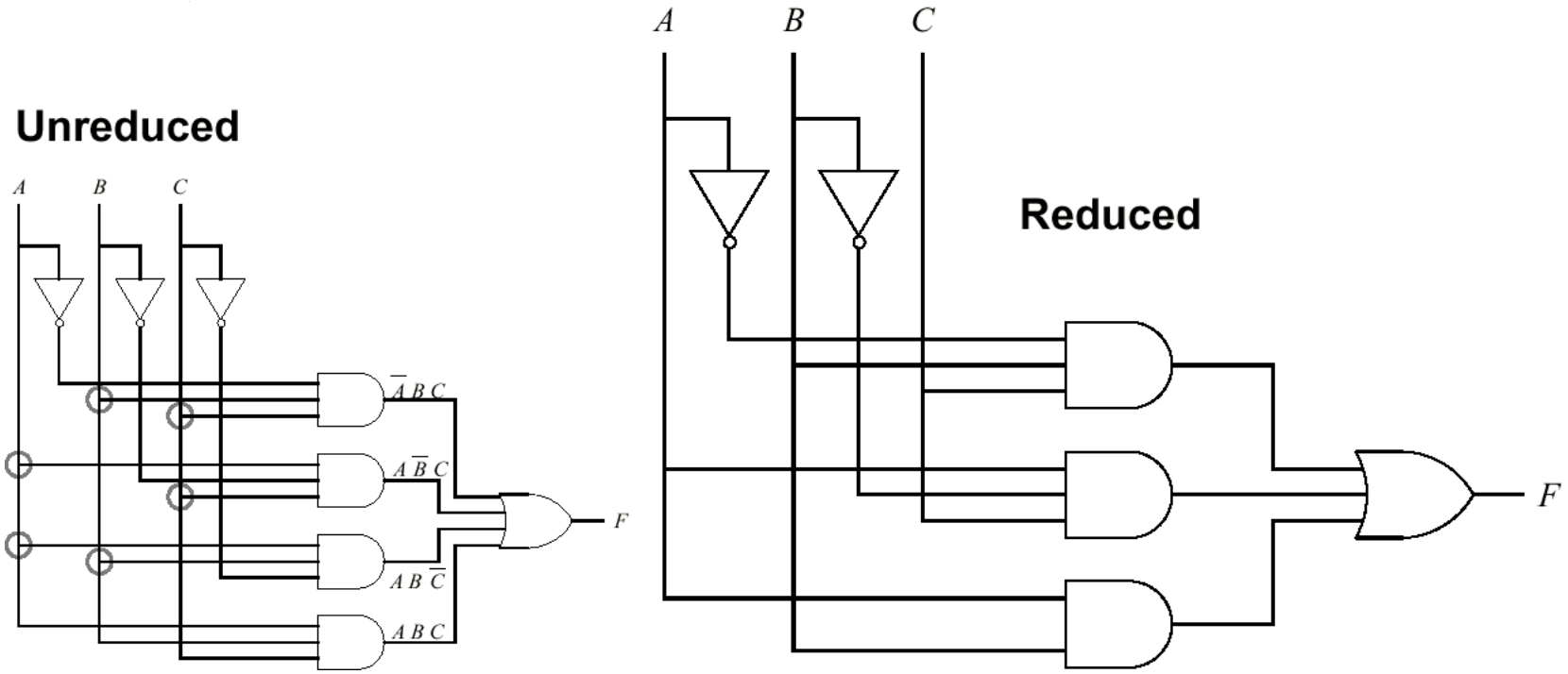
Reduction (Simplification) of Boolean Expressions

- It is usually possible to simplify the canonical SOP (or POS) forms.
- A smaller Boolean equation generally translates to a lower gate Count in the target circuit.
- We cover two methods: algebraic reduction, Karnaugh map reduction.
- A third method a tabular form is also used for machine driven reductions. This is called the Quine-McCluskey reduction technique.

© 1999 M. Murdocca and V. Heuring

Reduced Majority Function Circuit

- **Compared with the AND-OR circuit for the unreduced majority function, the inverter for C has been eliminated, one AND gate has been eliminated, and one AND gate has only two inputs instead of three inputs. Can the function by reduced further? How do we go about it?**



© 1999 M. Murdocca and V. Heuring

The Algebraic Method

- **Consider the majority function, *F*. We apply the algebraic method to reduce *F* to its minimal two-level form:**

$$F = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$F = \overline{A}BC + A\overline{B}C + AB(\overline{C} + C) \quad \text{Distributive Property}$$

$$F = \overline{A}BC + A\overline{B}C + AB(1) \quad \text{Complement Property}$$

$$F = \overline{A}BC + A\overline{B}C + AB \quad \text{Identity Property}$$

$$F = \overline{A}BC + A\overline{B}C + AB + ABC \quad \text{Idempotence}$$

$$F = \overline{A}BC + AC(\overline{B} + B) + AB \quad \text{Identity Property}$$

$$F = \overline{A}BC + AC + AB \quad \text{Complement and Identity}$$

$$F = \overline{A}BC + AC + AB + ABC \quad \text{Idempotence}$$

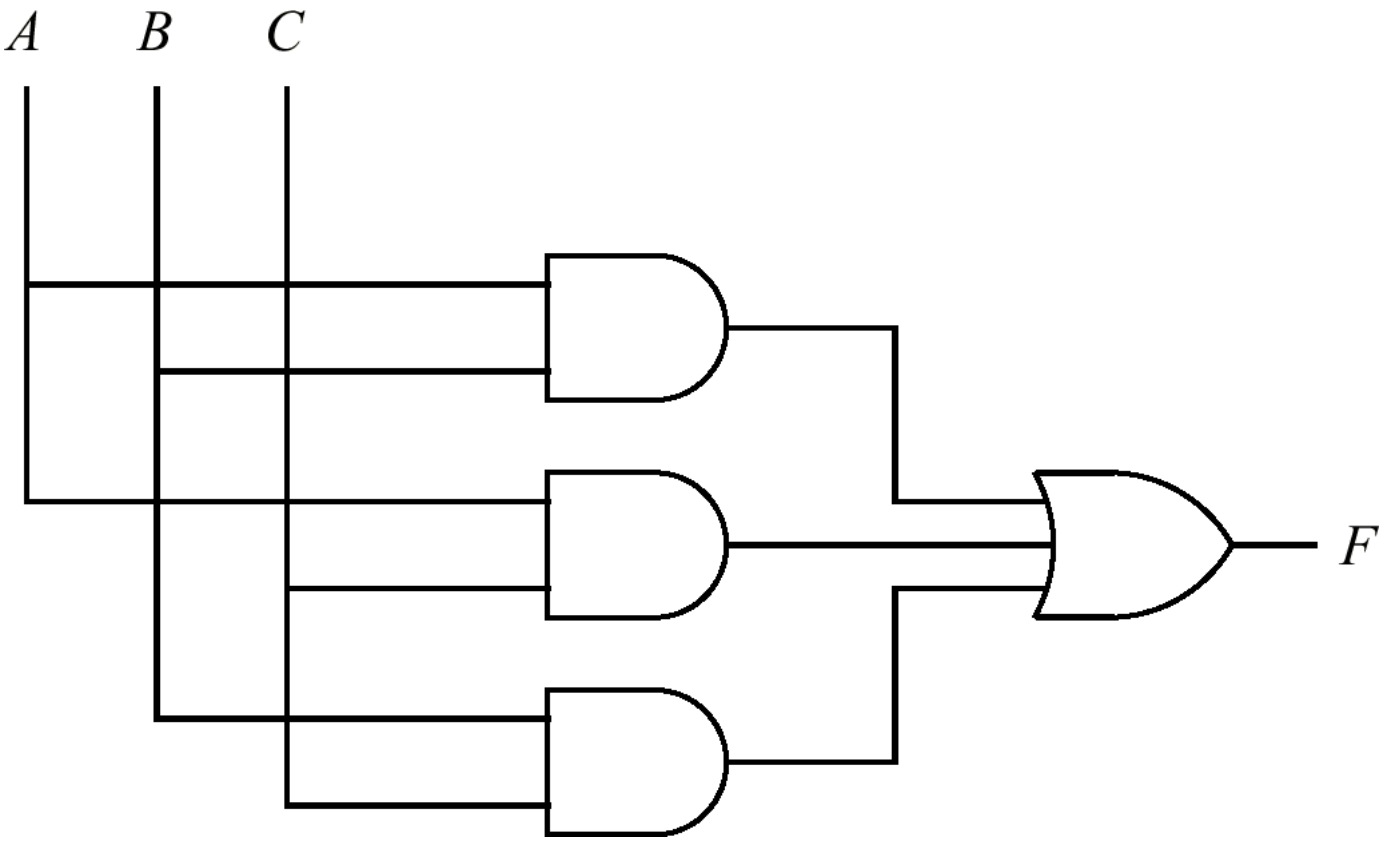
$$F = BC(\overline{A} + A) + AC + AB \quad \text{Distributive}$$

$$F = BC + AC + AB \quad \text{Complement and Identity}$$

© 1999 M. Murdocca and V. Heuring

The Algebraic Method

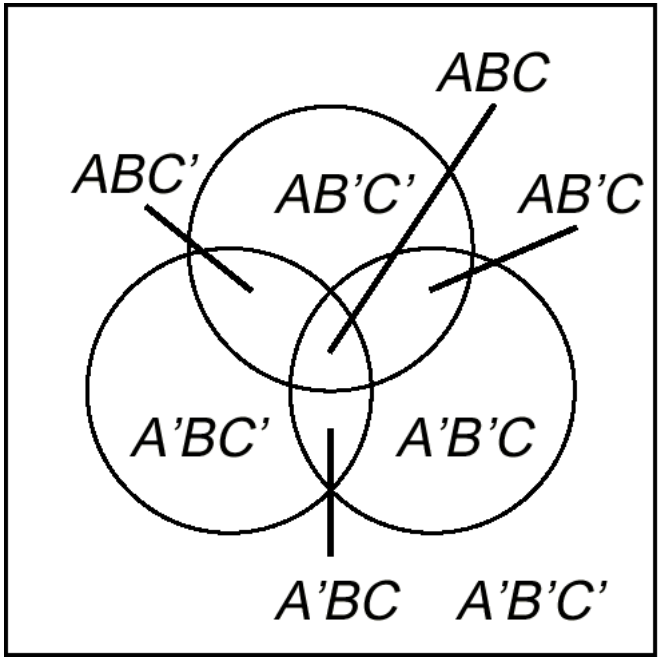
- **This majority circuit is functionally equivalent to the previous majority circuit, but this one is in its minimal two-level form:**

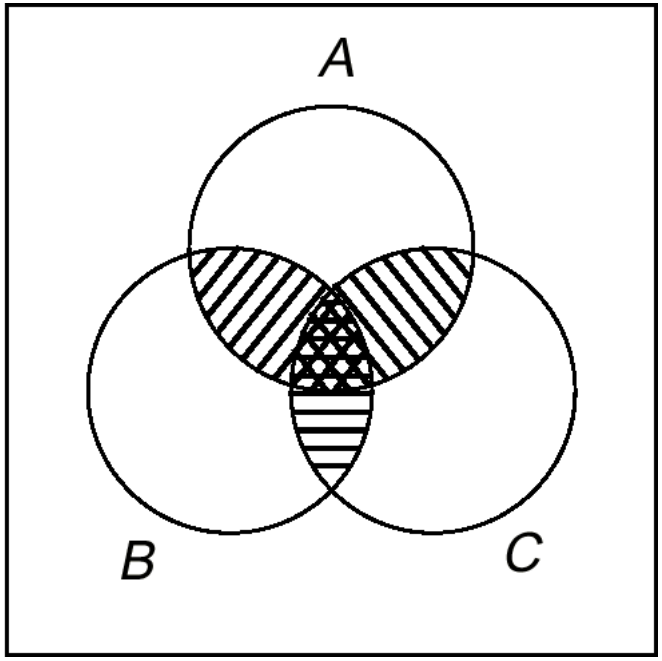


© 1999 M. Murdocca and V. Heuring

Karnaugh Maps: Venn Diagram Representation of Majority Function

- **Each distinct region in the "Universe" represents a minterm.**
- **This diagram can be transformed into a *Karnaugh Map*.**



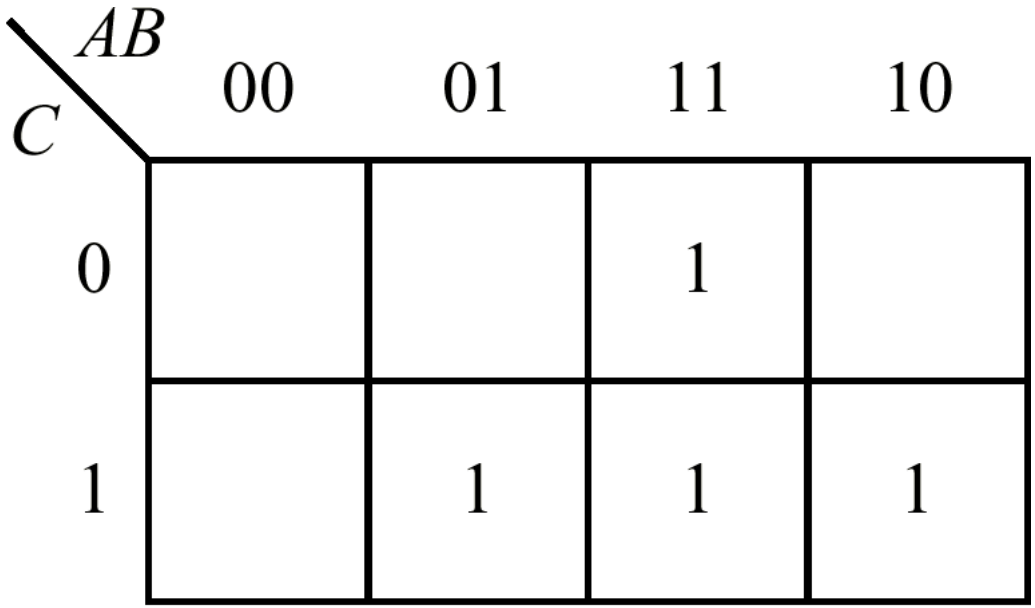


© 1999 M. Murdocca and V. Heuring

K-Map for Majority Function

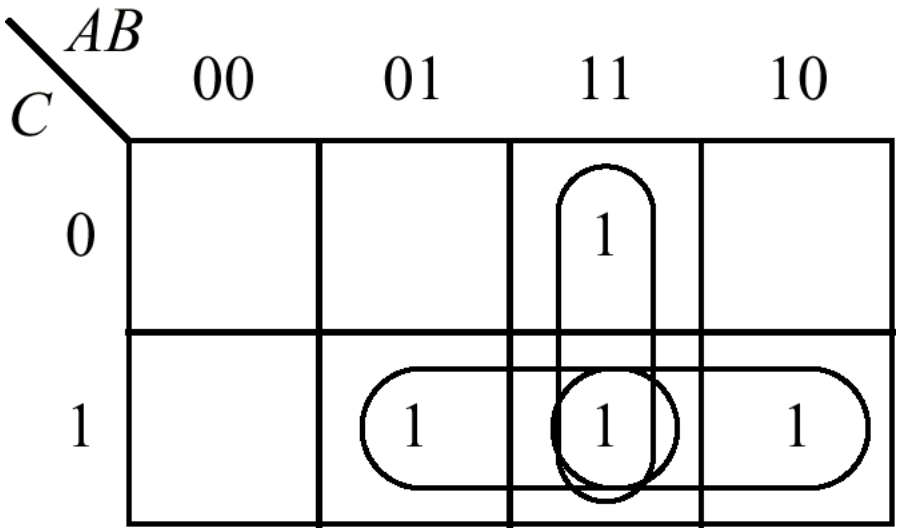
- **Place a “1” in each cell that corresponds to that minterm.**
- **Cells on the outer edge of the map “wrap around”**

Minterm Index	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



© 1999 M. Murdocca and V. Heuring

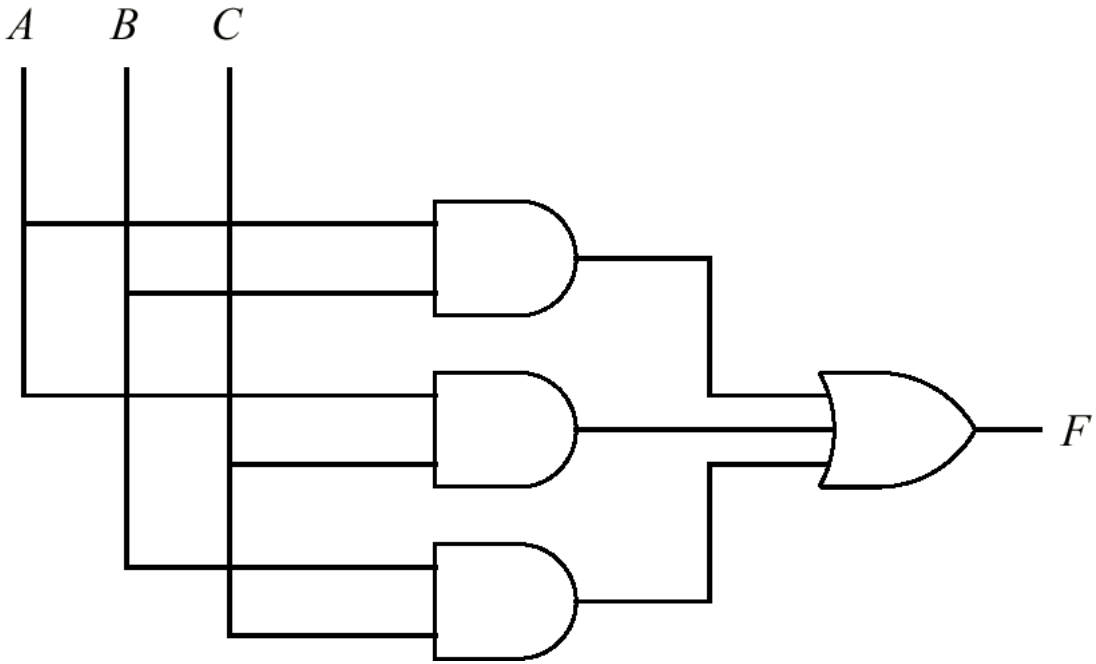
Adjacency Groupings for Majority Function



- **F = BC + AC + AB**

© 1999 M. Murdocca and V. Heuring

Minimized AND-OR Majority Circuit

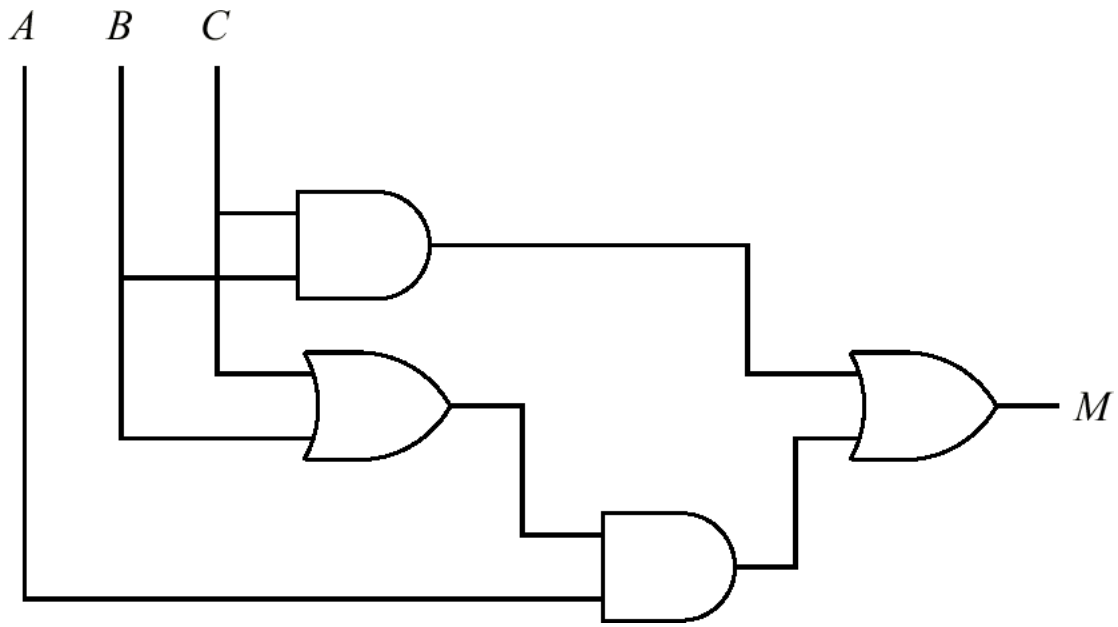


- F = BC + AC + AB
- The K-map approach yields the same minimal two-level form as the algebraic approach.

© 1999 M. Murdocca and V. Heuring

3-Level Majority Circuit

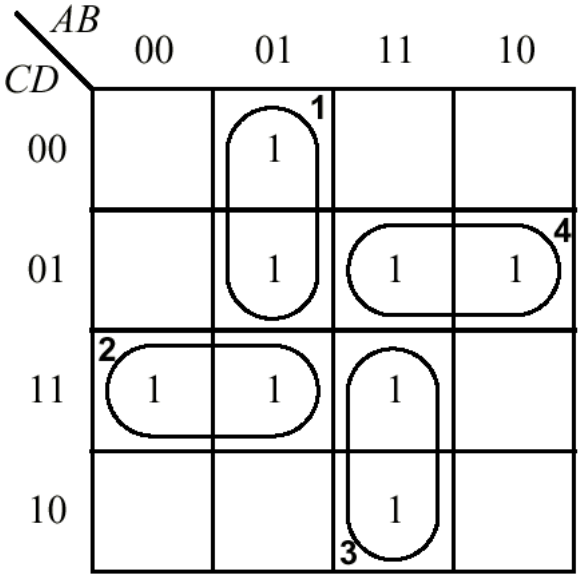
- **K-Kap Reduction results in a reduced two-level circuit (that is, AND followed by OR. Inverters are not included in the two-level count). Algebraic reduction can result in multi-level circuits with even fewer logic gates and fewer inputs to the logic gates.**



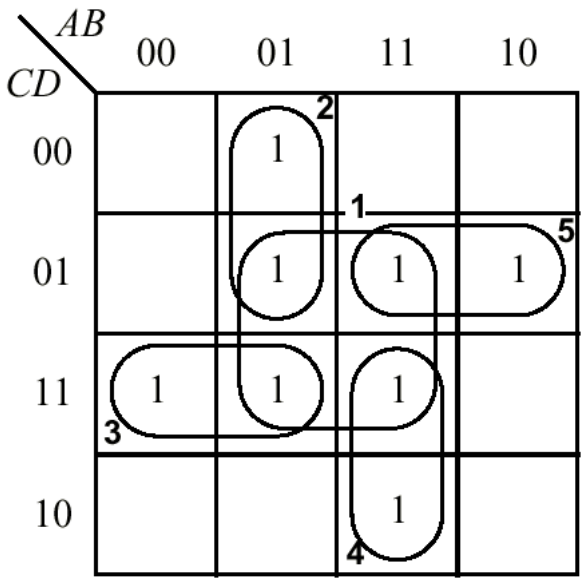
© 1999 M. Murdocca and V. Heuring

K-Map Groupings

- **Minimal grouping is on the left, non-minimal (but logically equivalent) grouping is on the right.**
- **To obtain minimal grouping, create *smallest* groups first.**



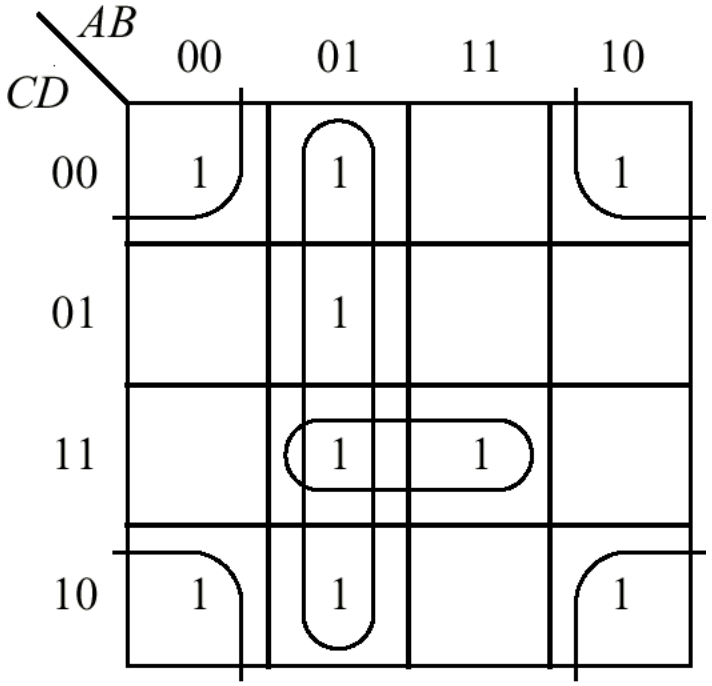
$$F = \overline{A}\,B\,\overline{C} + \overline{A}\,C\,D + A\,B\,C + A\,\overline{C}\,D$$



$$F = B\,D + \overline{A}\,B\,\overline{C} + \overline{A}\,C\,D + A\,B\,C + A\,\overline{C}\,D$$

© 1999 M. Murdocca and V. Heuring

K-Map Corners are Logically Adjacent



$$F = B\,C\,D + \overline{B}\,\overline{D} + \overline{A}\,B$$

© 1999 M. Murdocca and V. Heuring

Truth Table with Don't Cares

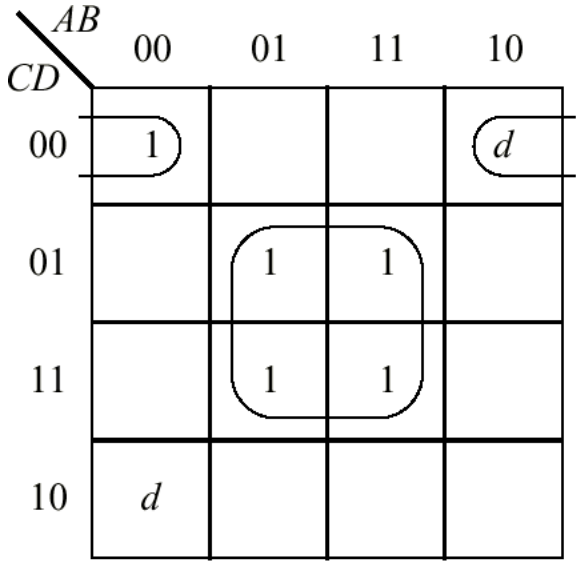
- **A truth table representation of a single function with don't cares.**

A	B	C	D	F
0	0	0	0	d
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	d
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	d

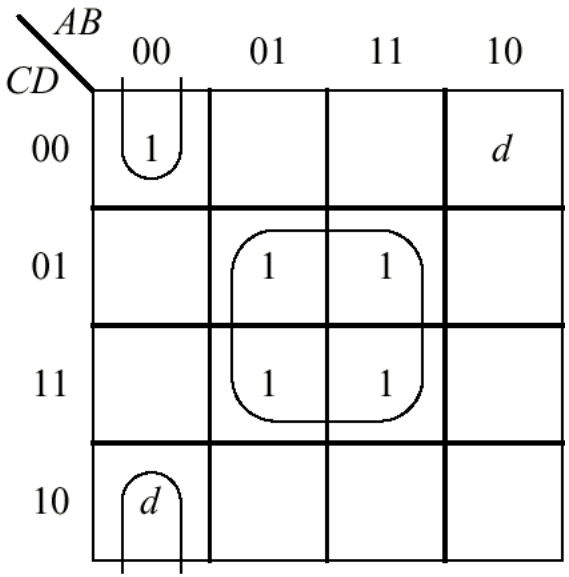
© 1999 M. Murdocca and V. Heuring

K-Maps and Don't Cares

- **There can be more than one minimal grouping, as a result of don't cares.**



$F = \overline{B}\,\overline{C}\,\overline{D} + B\,D$

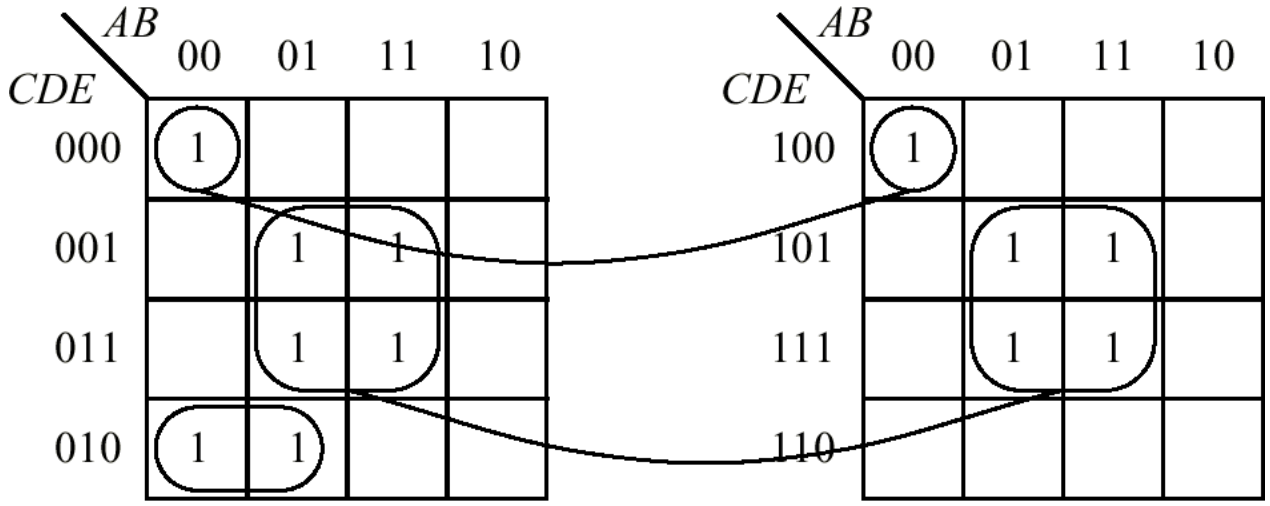


$F = \overline{A}\,\overline{B}\,\overline{D} + B\,D$

© 1999 M. Murdocca and V. Heuring

Five-Variable K-Map

- **Visualize two 4-variable K-maps stacked one on top of the other; groupings are made in three dimensional cubes.**

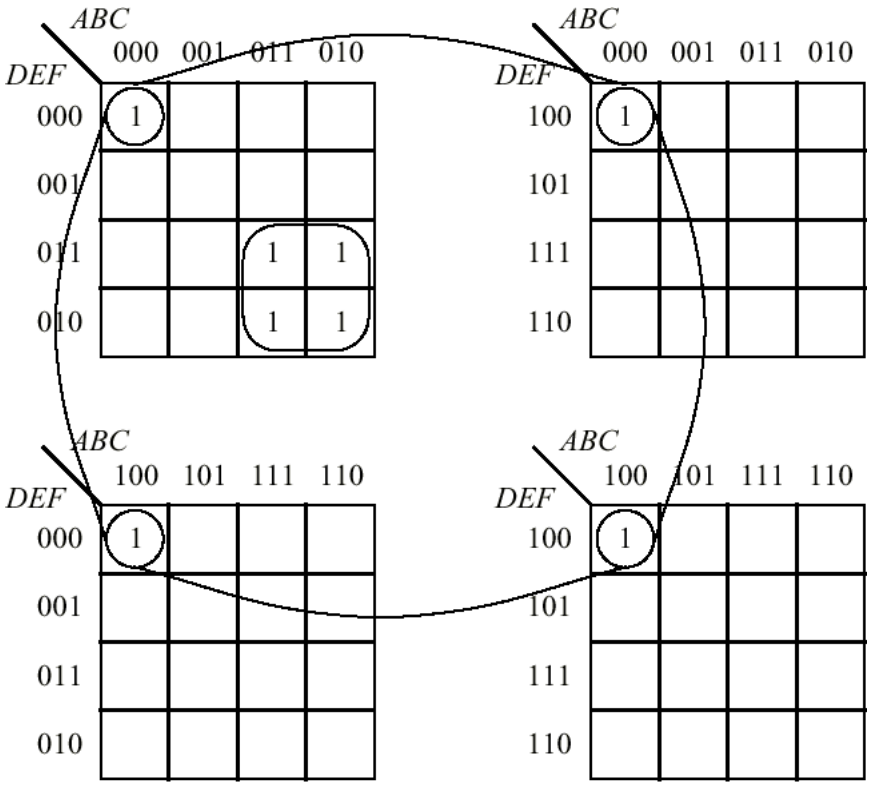


$$F = \overline{A}\,\overline{C}\,D\overline{E} + \overline{A}\,\overline{B}\,\overline{D}\,\overline{E} + B\,E$$

© 1999 M. Murdocca and V. Heuring

Six-Variable K-Map

- **Visualize four 4-variable K-maps stacked one on top of the other; groupings are made in three dimensional cubes.**

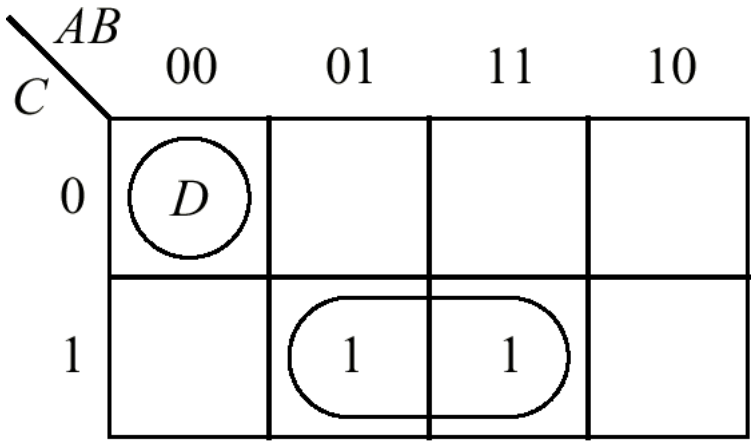


$$G = \overline{B}\,\overline{C}\,\overline{E}\,\overline{F} + \overline{A}\,B\,\overline{D}\,E$$

© 1999 M. Murdocca and V. Heuring

Map-Entered Variables

- **An example of a K-map with a map-entered variable *D*.**

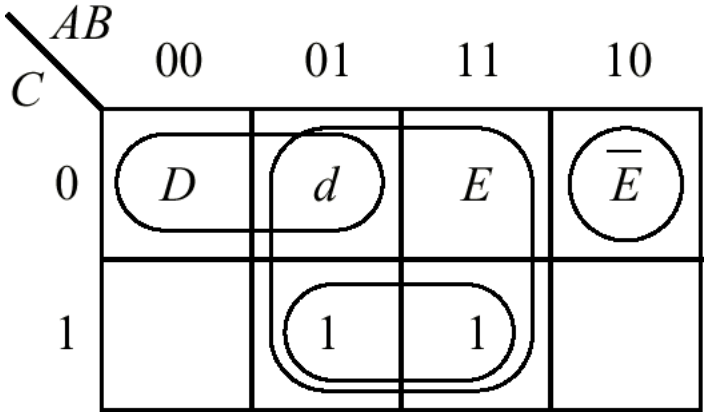


$$F = B\,C + \overline{A}\,\overline{B}\,\overline{C}\,D$$

© 1999 M. Murdocca and V. Heuring

Two Map-Entered Variables

- **A K-map with two map-entered variables *D* and *E*.**
- ***F = BC + ACD + BE + ABCE***



© 1999 M. Murdocca and V. Heuring

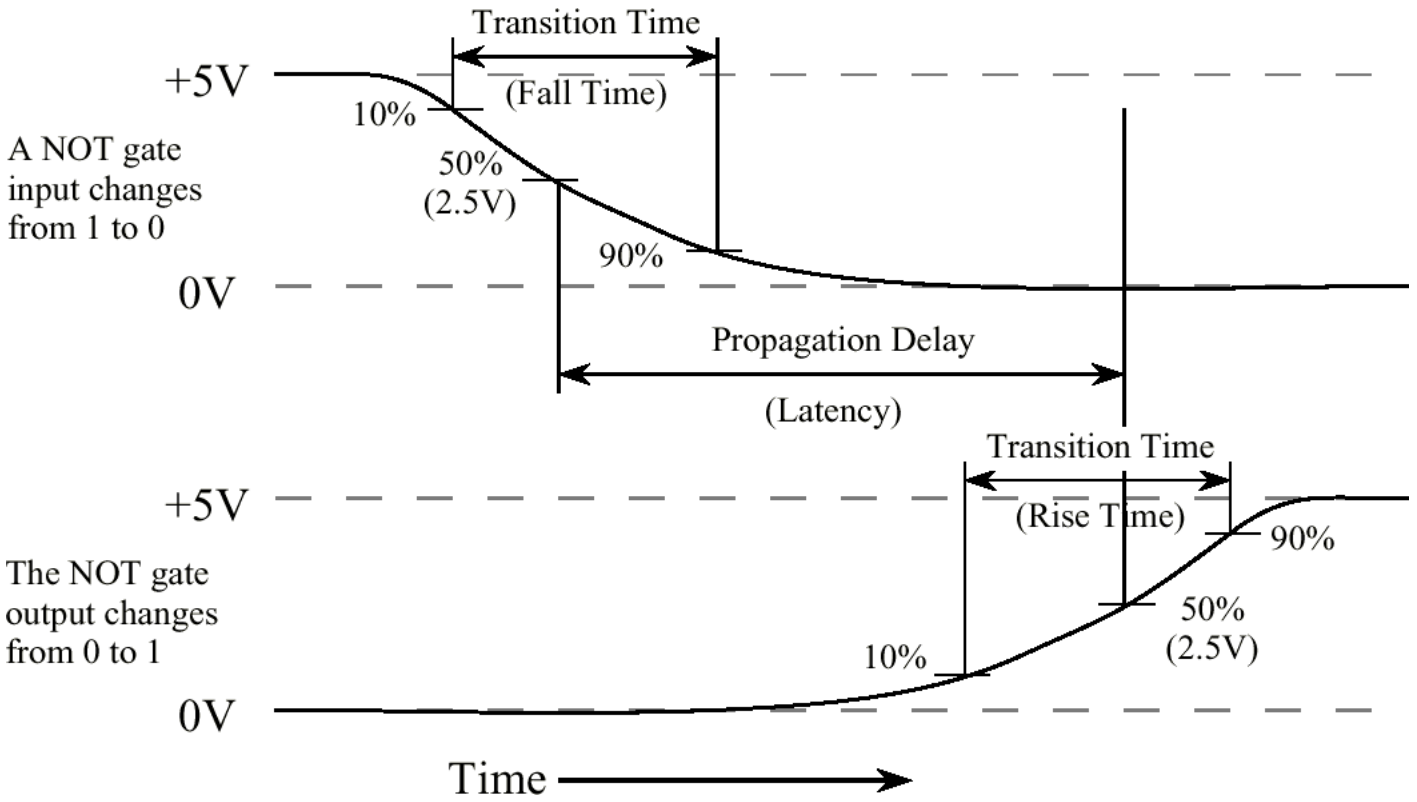
Speed and Performance

- The speed of a digital system is governed by:
  - the propagation delay through the logic gates and
  - the propagation delay across interconnections.
- We will look at characterizing the delay for a logic gate, and a method of reducing circuit depth using function decomposition.

© 1999 M. Murdocca and V. Heuring

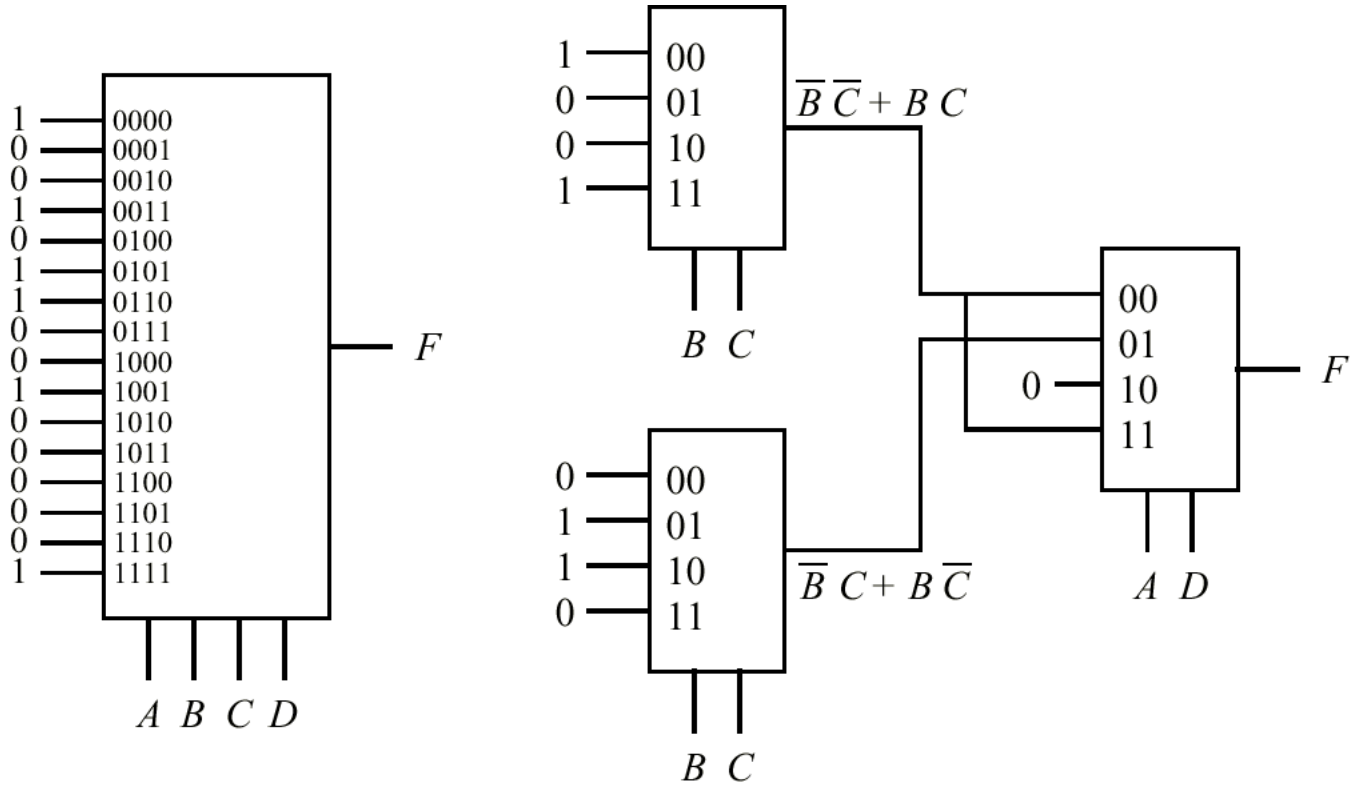
Propagation Delay for a NOT Gate

- **(From Hamacher *et. al.* 1990)**



© 1999 M. Murdocca and V. Heuring

MUX Decomposition



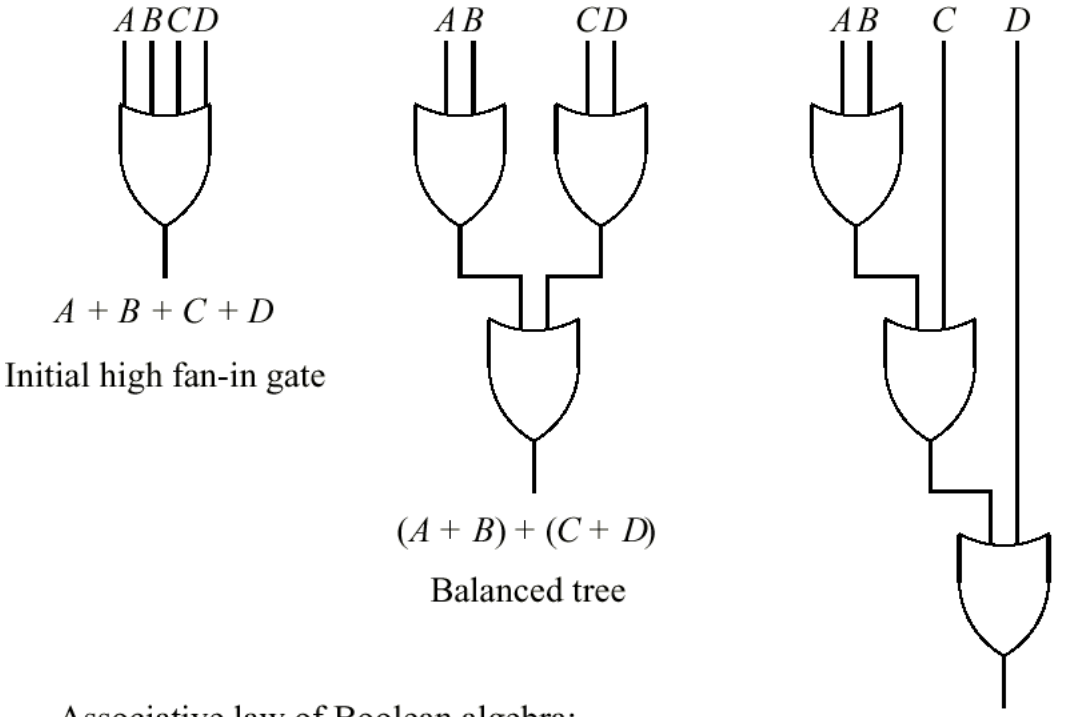
$$F(ABCD) = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,CD + \overline{A}\,B\overline{C}\,D + \overline{A}\,BC\overline{D} + A\overline{B}\,\overline{C}\,D + ABCD$$

$$= (\overline{B}\,\overline{C} + BC)AD + (\overline{B}\,C + B\overline{C})\overline{A}\,D + (\overline{B}\,\overline{C} + BC)$$

© 1999 M. Murdocca and V. Heuring

OR-Gate Decomposition

- **Fanin affects circuit depth.**



Associative law of Boolean algebra:

$$A + B + C + D = (A + B) + (C + D)$$

© 1999 M. Murdocca and V. Heuring

Objectives Completed

- Explained the need for logic reduction
- Name 3 methods of logic reduction
- Reduced logic expressions using
  - Boolean algebra theorems
  - Karnaugh-Maps
- Characterize propagation delays for a gate
- Use OR gate, MUX decomposition to simplify logic circuits

SSI : small scale integrated circuit

Next time we will

- Examine the carry lookahead adder
- Learn to run Digisim