



University of
Massachusetts
Amherst

Engin112 – Lecture 14-15

Combinational Circuits

Maciej Ciesielski
Department of Electrical and Computer Engineering
10/07-11/2011

Recap from Last Lecture

- **Karnaugh maps**
 - Incompletely specified functions
 - Essential prime implicants
 - 4- and 5-variable maps
 - Converting maps to expressions
 - Don't care conditions
 - » Use unspecified outputs to increase prime implicants and minimize logic

- **Today's lecture**
 - Implementing functions with one type of gate
 - » NAND
 - » NOR

NAND Implementations

- Any logic circuit can be built using single “universal” gate
 - NAND or NOR

integrated circuit datasheet

FAIRCHILD SEMICONDUCTOR™

October 1987
Revised May 2002

MM74C00 • MM74C02 • MM74C04
Quad 2-Input NAND Gate •
Quad 2-Input NOR Gate •
Hex Inverter

General Description
 The MM74C00, MM74C02, and MM74C04 logic gates employ complementary MOS (CMOS) to achieve wide power supply operating range, low power consumption, high noise immunity and symmetric controlled rise and fall times. With features such as this the 74C logic family is close to ideal for use in digital systems. Function and pin out compatibility with series 74 devices minimizes design time for those designers already familiar with the standard 74 logic family.
 All inputs are protected from damage due to static discharge by diode clamps to V_{CC} and GND.

Features

- Wide supply voltage range: 3V to 15V
- Guaranteed noise margin: 1V
- High noise immunity: $0.45 V_{CC}$ (typ.)
- Low power consumption: 10 nW/package (typ.)
- Low power: TTL compatibility: Fan out of 2 driving 74L

Connection Diagrams

MM74C00

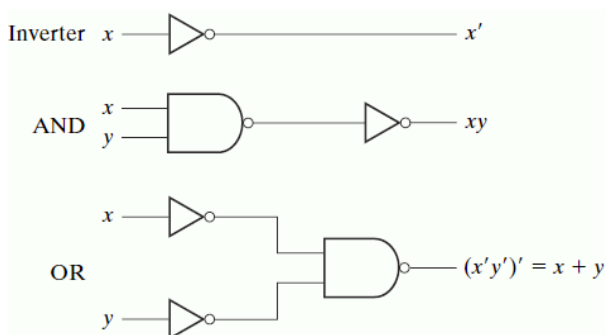
 Top View

MM74C02

 Top View

Universality of NAND

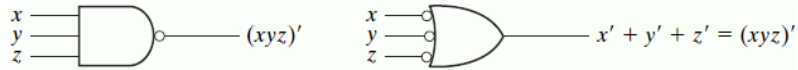
- NOT, AND, and OR can be implemented with NAND



- Single-input NAND is inverter

Conversion to NAND Implementation

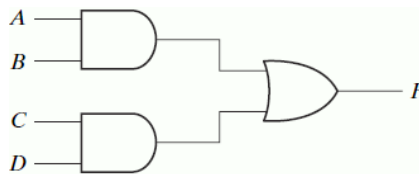
- Minimized expressions are AND-OR combinations
 - Two ways to represent NAND
 - AND-invert
 - Invert-OR (use De Morgan law)



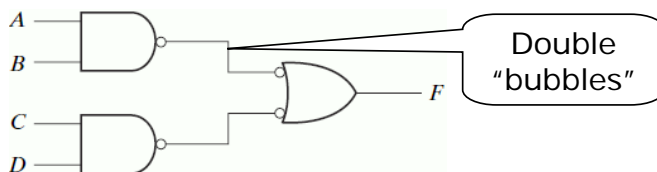
- Key observation: two “bubbles” eliminate each other
 - Two bubbles — equal straight wire
- How to generate a sum of minterms using NAND?
 - Use AND-invert for minterms
 - Use invert-OR (OR with inverted inputs) for sum

Conversion to NAND Implementation

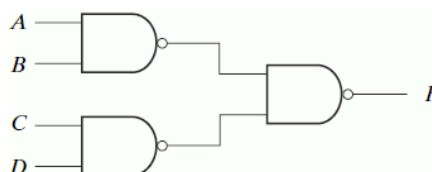
- Sum of minterms



- Replace AND with AND-invert and OR with inverted inputs
 - Still same circuit!

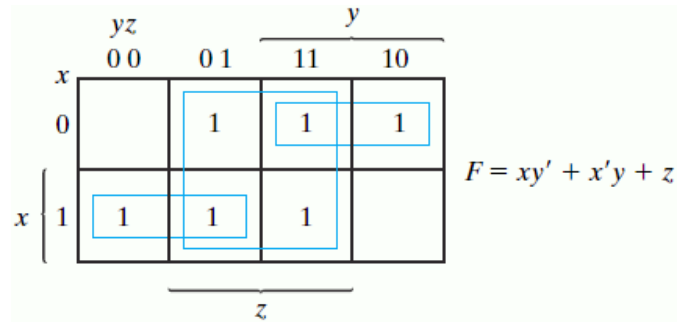


- Replace AND-invert and invert-OR with NAND

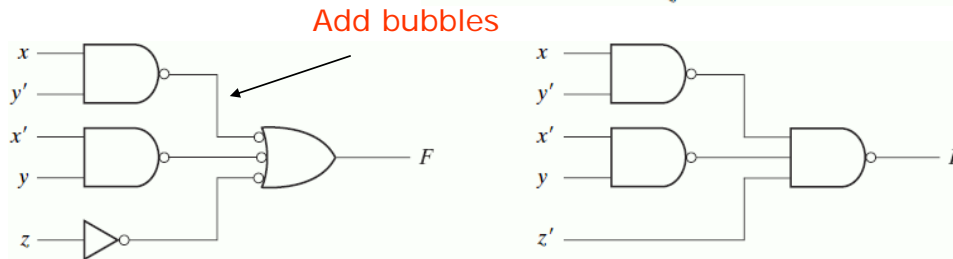


NAND Example

- Function $F = \sum(1,2,3,4,5,7)$
 - Minimize and implement with NAND
- Karnaugh map:



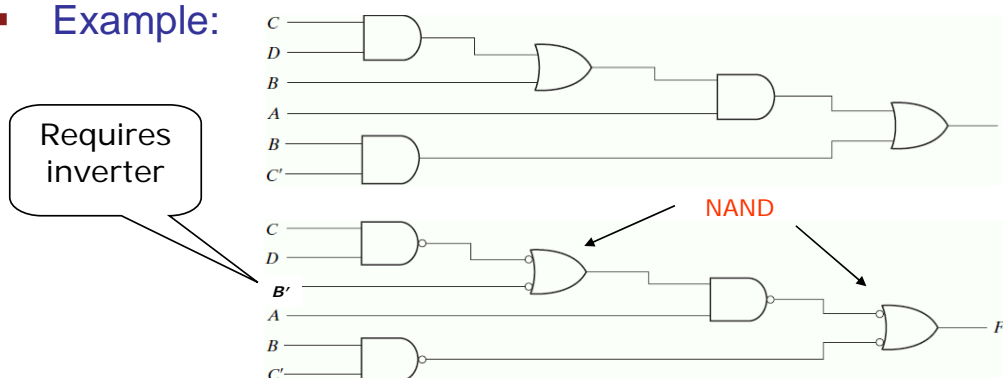
- Implementation:



Multilevel NAND circuits

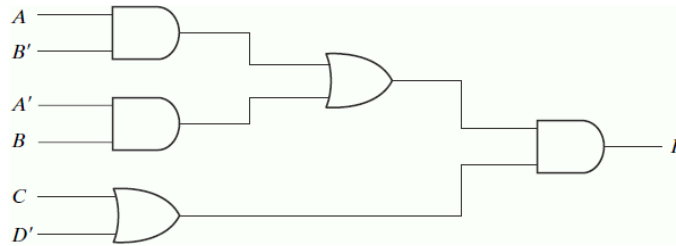
- Multilevel circuits conversion rules:
 - Convert all AND gates to NAND with AND-invert symbols
 - Convert all OR gates to NAND with invert-OR symbols
 - Check all bubbles in diagram. For every bubble that is not compensated by another bubble, insert inverter

- Example:

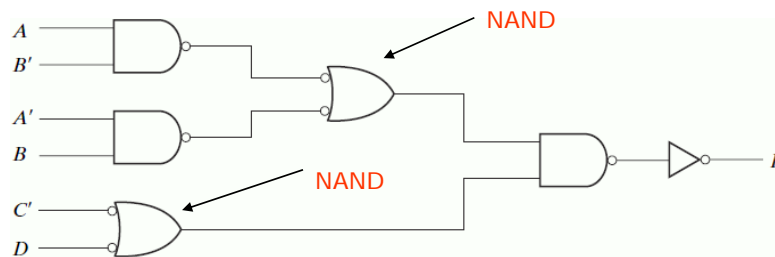


Multilevel NAND Circuits

- $F = (AB' + A'B)(C + D')$

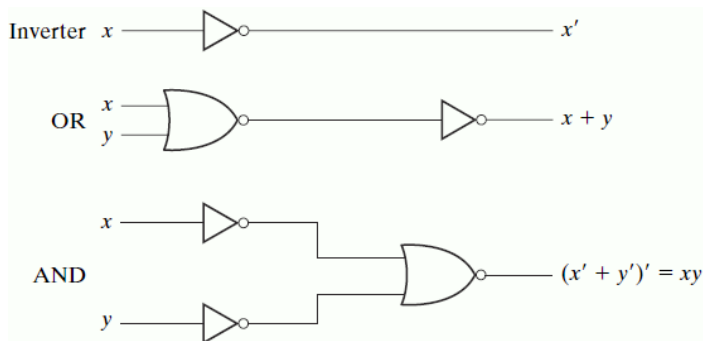


- With NAND gates:

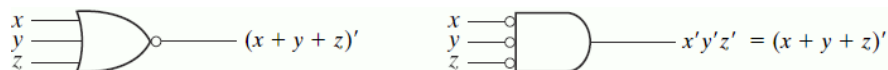


NOR Implementations

- NOR can also replace NOT, AND, OR:

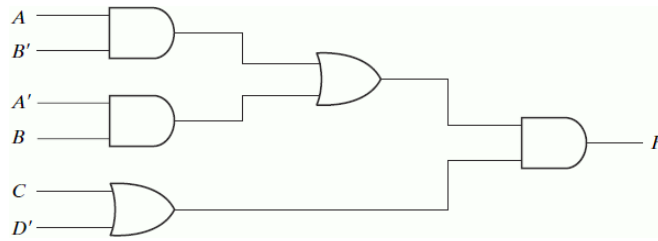


- Two representations of NOR:
 - OR-invert and invert-AND (AND with inverted inputs)

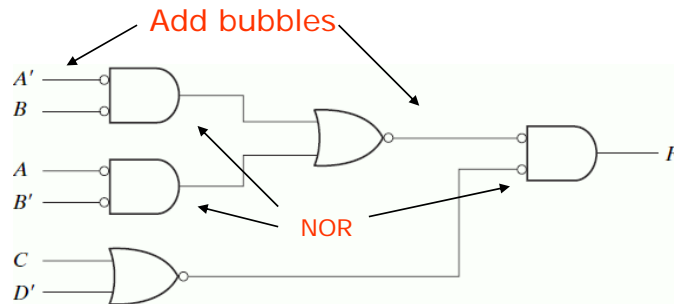


Converting to NOR Implementations

- Same rules as for NAND implementations
- $F = (AB' + A'B)(C + D')$



- With NOR:



Conversion Example

- Contest: implement function
 - $F(A,B,C,D) = \sum(0,2,7,8,10,13)$
 - $d(A,B,C,D) = \sum(6,14,15)$
 - Group 1: K-map, minimize function, NAND implementation
 - Group 2: K-map, minimize function, NOR implementation
- Solution (SOP form):

$$F = BC + B'D' + ABD$$

		<i>CD</i>			
		00	01	11	10
<i>AB</i>	00				
	01				
	11				
	10				

Conversion Example

- Contest: implement function

$$F(A,B,C,D) = \sum(0,2,7,8,10,13)$$

$$d(A,B,C,D) = \sum(6,14,15)$$

- Group 1: K-map, minimize function, NAND implementation
- Group 2: K-map, minimize function, NOR implementation
- Solution (SOP form): $F = BC + B'D' + ABD$

NAND

$$F = [(BC + B'D' + ABD)']'$$

$$= [(BC)' \cdot (B'D')' \cdot (ABD)']'$$

NOR

$$F = \{ [(BC)'' + (B'D')'' + (ABD)'']' \}'$$

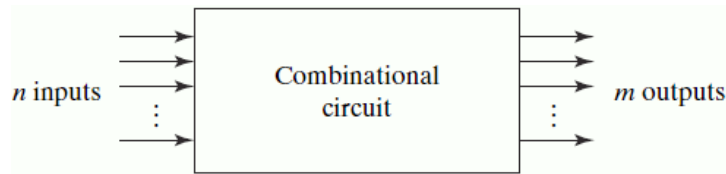
$$= \{ [(B'+C)'] + (B+D)'] + (A'+B'+D)']' \}'$$

Recap from last lectures

- Incompletely specified functions
 - Don't cares in Karnaugh map
- NAND and NOR implementations
 - NOT, AND, OR from NAND or OR gates
 - Convert sum of terms into NAND implementation
 - AND-invert and invert-OR representation
 - "Double bubbles"
- This lecture
 - Combinational logic circuits
 - Analysis of combinational circuits
 - Design of combinational circuits

Combinational Circuits

- So far: single function with one output
 - Most realistic problems use multiple outputs
- Need to extend to combinational circuits:



- **Combinational circuits**
 - » Outputs depend on the inputs (only)
 - » Special class of combinational circuits: arithmetic (next lecture)
- Important distinction: combinational vs sequential circuits
 - **Sequential circuits** (next chapter)
 - » Outputs depend on inputs and previous state (outputs)
 - » Use memory to store the "state" of design, feedback
 - » Previous inputs have effect on output

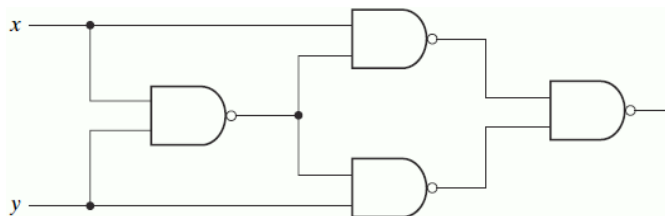
10/07-11/2011

Engin 112 - Intro to ECE

15

Analysis of Combinational Circuits

- Determine the function of circuit
 - Instead of developing the circuit based on the function
- Circuit analysis
 - Determine the output functions as algebraic expressions
 - Determine the truth table of the outputs
- What is the output function of this circuit?



10/07-11/2011

Engin 112 - Intro to ECE

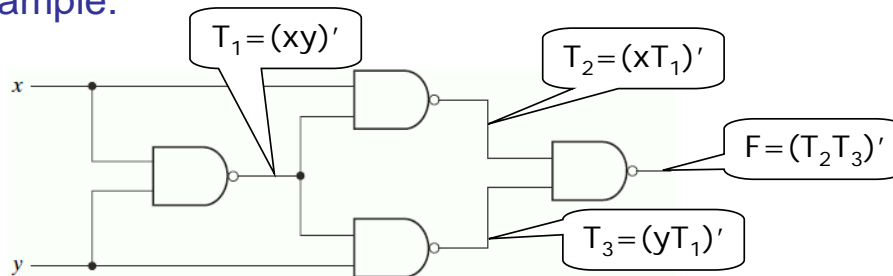
16

Circuit Analysis

- Analysis steps

1. Label all gate outputs with symbols
2. Determine Boolean function at the output of each gate
3. Express functions in terms of input variables + simplify

- Example:



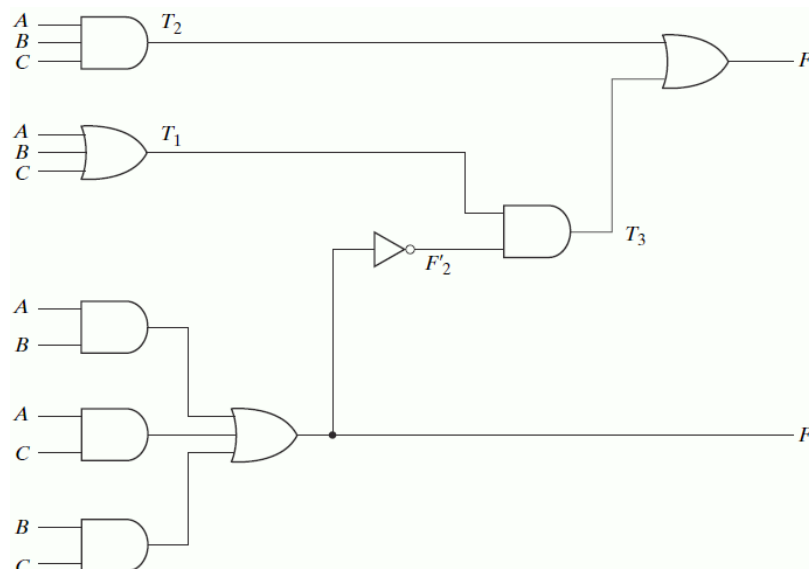
- Substitution:

- $$F = (T_2T_3)' = ((xT_1)'(yT_1)')' = (xT_1) + (yT_1) = x(xy)' + y(xy)' =$$

$$= (x(x'+y')) + (y(x'+y')) = xx' + xy' + yx' + yy' = xy' + yx' = x \oplus y$$

Circuit analysis Example 1

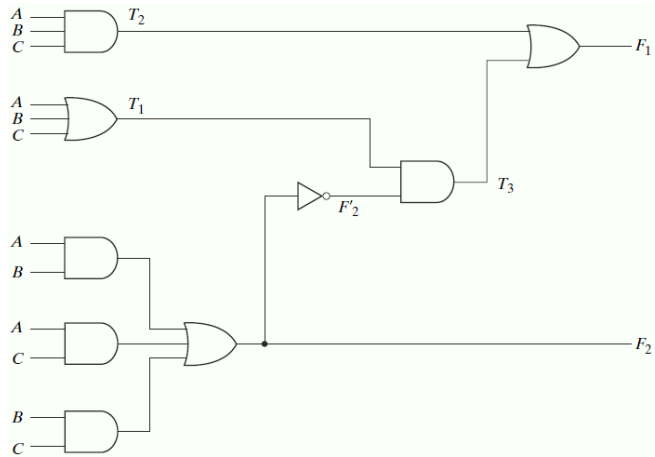
- What are the output functions F_1 and F_2 ?



Circuit Analysis Example 1, cont'd

1. Start with expressions that depend only on *input* variables:

- $T_2 = ABC$
- $T_1 = A+B+C$
- $F_2 = AB + AC + BC$



2. Express other outputs that depend on already defined internal signals

- $T_3 = F_2' T_1$
- $F_1 = T_3 + T_2$

10/07-11/2011

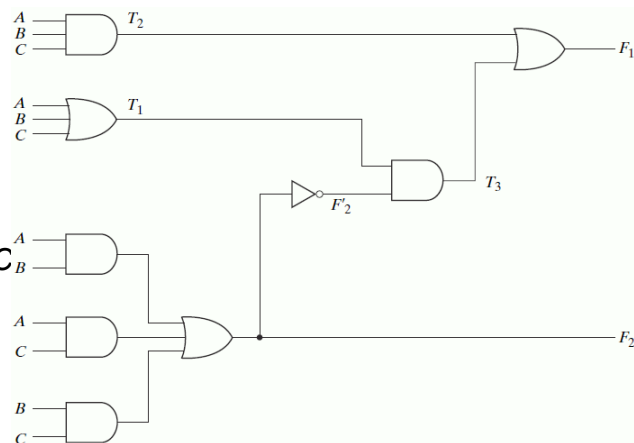
Engin 112 - Intro to ECE

19

Circuit Analysis Example 1, cont'd

3. Express outputs in terms of primary inputs and simplify:

$$\begin{aligned}
 F_1 &= T_3 + T_2 \\
 &= F_2' T_1 + ABC \\
 &= (AB + AC + BC)' (A + B + C) + ABC \\
 &= (A' + B') (A' + C') (B' + C') (A + B + C) + ABC \\
 &= (A' A' + B' A' + A' C' + B' C') \\
 &\quad (B' A + C' A + B' B + C' B + B' C + C' C) \\
 &\quad + ABC \\
 &= (A' + B' C') (AB' + AC' + BC' + B' C) \\
 &\quad + ABC \\
 &= A' AB' + A' AC' + A' BC' + A' B' C + AB' C' + \\
 &\quad AB' C' + BB' C' + B' CC' + ABC \\
 &= A' BC' + A' B' C + AB' C' + ABC
 \end{aligned}$$



10/07-11/2011

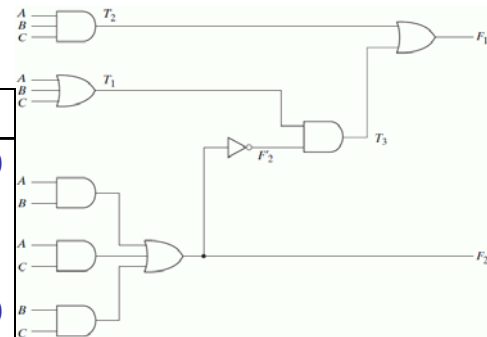
Engin 112 - Intro to ECE

20

Analysis by Truth Table

- Truth table:

A	B	C	T2	T1	F2	F2'	T3	F1
0	0	0	0	0	0	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	1	0	0	0
1	0	0	0	1	0	1	1	1
1	0	1	0	1	1	0	0	0
1	1	0	0	1	1	0	0	0
1	1	1	1	1	1	0	0	1



Design of Combinational Circuit

- Design procedure:
 1. From specification, determine required inputs and outputs
 2. Derive truth table
 3. Obtain simplified Boolean functions (one for each output)
 4. Draw logic diagram and verify correctness
- Often multiple possible solutions
 - Secondary criteria can determine choice
 - » Number of gates (products, sums)
 - » Number of connections (literals)
 - » Propagation time of signal
 - » Limits on number of outputs on gate ("driving capability")
 - Simplified Boolean expressions are typically OK

Design Example 2

- Design specification:
 - Develop a circuit that converts a BCD digit into Excess-3 code

- Step 1: inputs and outputs

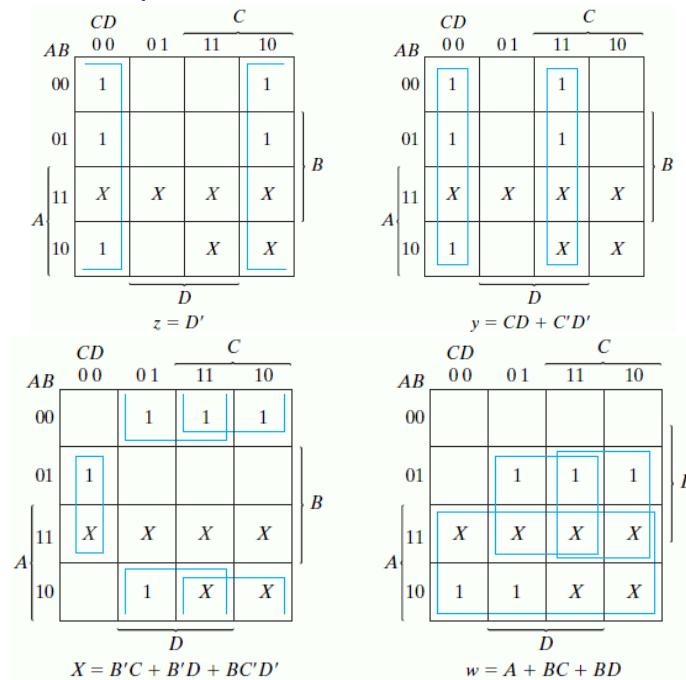
- Input: BCD digit
 - » 4 inputs: A, B, C, D
- Output: Excess-3 digit
 - » 4 outputs: w, x, y, z

- Step 2: truth table

A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

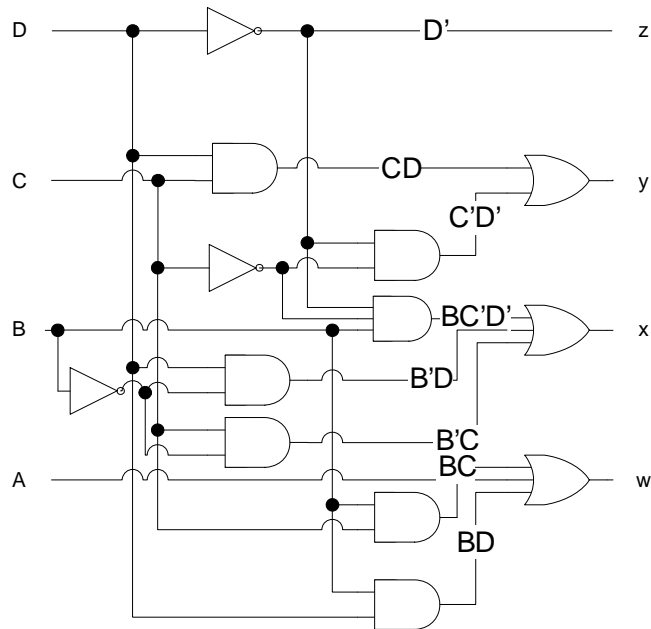
Design Example 2, cont'd

- Step 3: minimize output functions



Design Example 2, cont'd

- Step 4: circuit diagram (7 AND, 3 OR, 3 NOT gates)



10/07-11/2011

Engin 112 - Intro to ECE

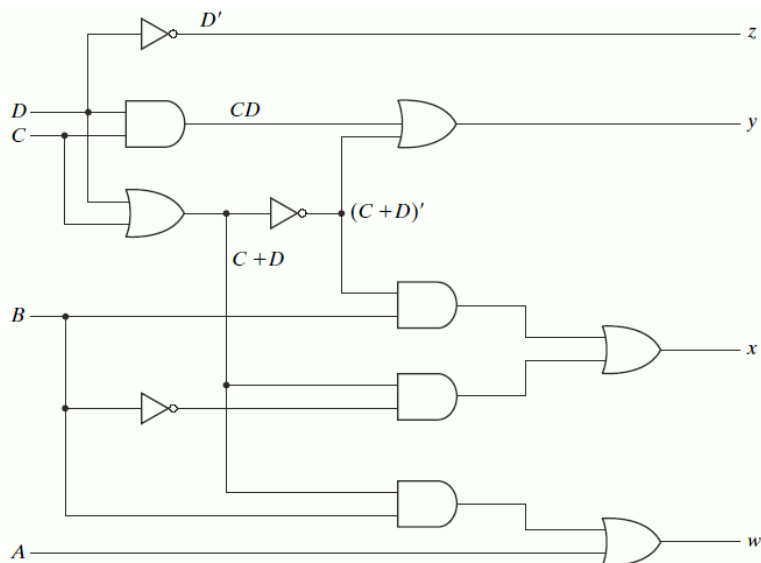
25

Design Example 2, cont'd

- Alternate solution

- Simplification:

- $z = D'$
- $y = CD + C'D'$
 $= CD + (C+D)'$
- $x = B'C + B'D + BC'D'$
 $= B'(C+D) + BC'D'$
 $= B'(C+D) + B(C+D)'$
- $w = A + BC + BD$
 $= A + B(C+D)$



- What's the difference?

- 4 AND, 4 OR, 2 NOT gates,
vs
- 7 AND, 3 OR, 3 NOT gates

10/07-11/2011

Engin 112 - Intro to ECE

26

Summary (Chapter 3 + 4.1-4.4)

- Learned about analyzing, constructing and minimizing logic functions
- Canonical forms
 - Sum of minterms (truth table)
 - Product of maxterms
- Standard forms
 - Sum of Products, Product of Sums
- Karnaugh maps for logic minimization
 - Completely specified logic functions
 - Incompletely specified logic functions (with don't cares)
- NAND and NOR implementations
 - Converting between the two forms
- Combinational circuits
 - Analysis
 - Design (synthesis)