Analysis and Design of Combinational Circuit, Half Adder, Full Adder, Adder/Subtractor

Instructor: Engr. Rimsha

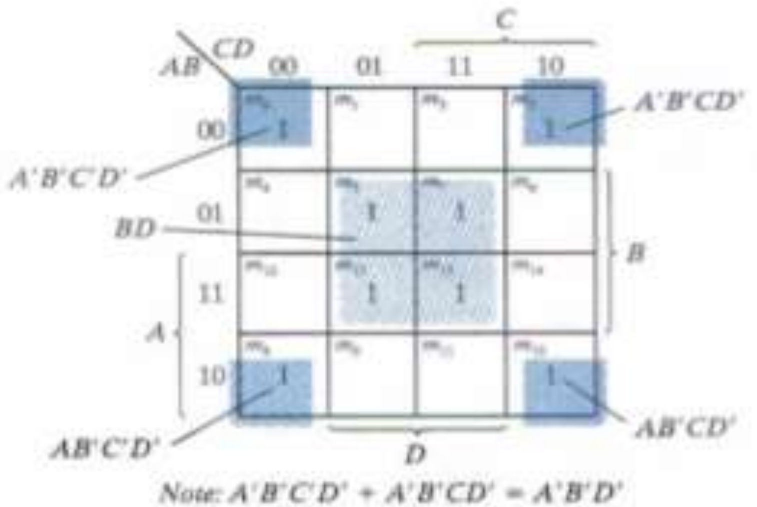
Prime Implicants

$$F(A, B, C, D) = \Sigma(0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$$

Prime implicant is a product term obtained by combining the maximum possible number of adjacent squares in the map

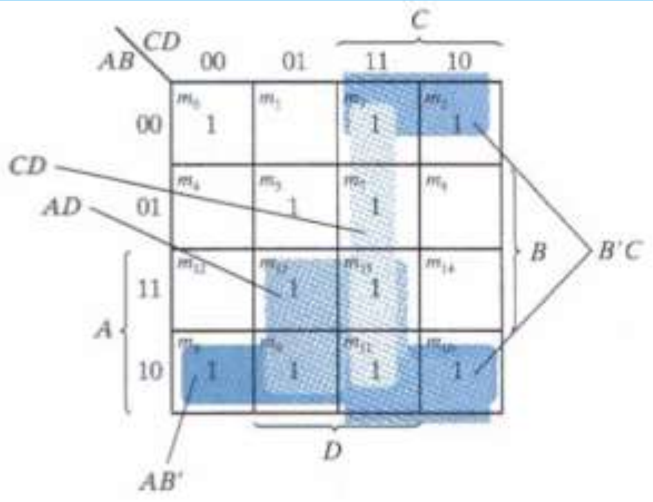
If a mintem in a square is covered by only one prime implicant, that prime implicant is said to be ***Essential Prime Implicant***

Essential Prime Implicant



Note: $A'B'C'D' + A'B'CD' = A'B'D'$
$AB'C'D' + AB'CD' = AB'D'$
$A'B'D' + AB'D' = B'D'$

(a) Essential prime implicants BD and $B'D'$



(b) Prime implicants CD, $B'C$, AD, and AB'

$$
\begin{aligned}
F &= BD + B'D' + CD + AD \\
&= BD + B'D' + CD + AB' \\
&= BD + B'D' + B'C + AD \\
&= BD + B'D' + B'C + AB'
\end{aligned}
$$

Essential Prime Implicant Example

G(W,X,Y,Z) $= \Sigma m(0, 1, 3, 7, 8, 11, 12, 13, 15)$

$$F = YZ + W'X'Y' + X'Y'Z' + WXY'$$

$$F = YZ + W'X'Y' + WY'Z' + WXY'$$

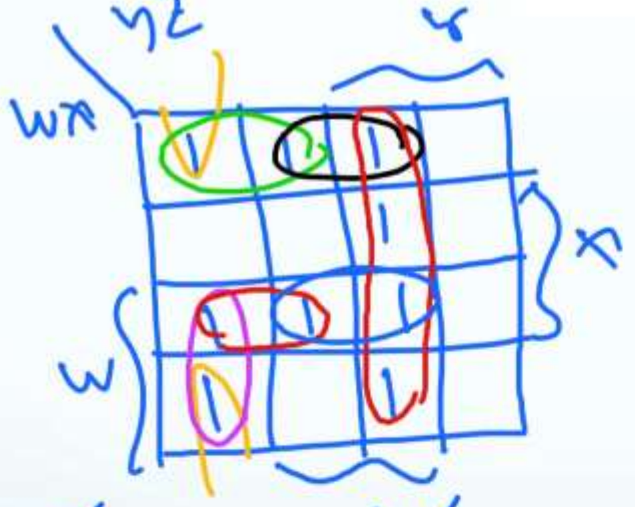
$$F = YZ + W'X'Y' + WY'Z' + WXZ$$

✓

$F_1 = yz + x'y'z' + w'x'y + wxz$

Essential Prime Implicant Example

G(W,X,Y,Z) $= \Sigma m(0, 1, 3, 7, 8, 11, 12, 13, 15)$



$(3,7,11,15)$

Essential = yz

PIs = $x'y'z'$, $w'x'y'$, $w'x'z$, $wy'z'$, wxy', wxz

Patrick Method.

	Prime Impl.	Comb.	0	1			8		12	13	
*	yz	(3,7,11,15)									
A	$x'y'z'$	(0,8)	X				X				
B	$w'x'y'$	0,1	X	X							
C	$w'x'z$	1,3		X							
D	$wy'z'$	8,12					X		X		
E	wxy'	12,13							X	X	
F	wxz	13,15								X	

✓ 2 ✓

$$G = (A+B)(B+C)(A+D)(D+E)(E+F)$$

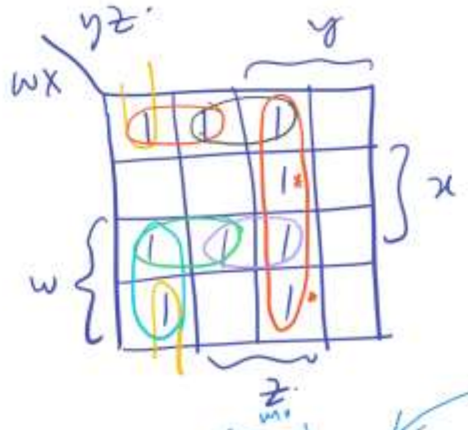
$$= (A+BD)(B+C)(E+DF)$$

$$= [AB + AC + BD + BCD][E+DF]$$

$$= \boxed{ABE} + ACE + BDE + \cancel{BCDE} + \cancel{ABDF} + \cancel{ACDF} + BDF + \cancel{BCDF}$$

Essential Prime Implicant Example Solution

G(W,X,Y,Z) $= \Sigma m(0, 1, 3, 7, 8, 11, 12, 13, 15)$



EPI yz

PI $x'y'z'$, $w'x'y'$, $wy'z'$, $w'x'z$

m_0, m_8 m_0, m_1 m_8, m_{12} m_1, m_3

0,1,8,12,13

wxy' wxz

m_{12}, m_{13} m_{13}, m_{15}

① $F = yz + \overset{0,8}{x'y'z'} + \overset{1,3}{w'x'z} + \overset{12,13}{wxy'}$

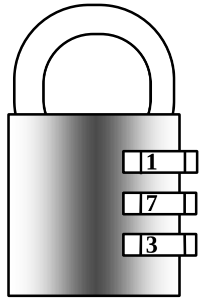
✓ ② $F = yz + \underset{0,8}{x'y'z'} + \underset{0,1}{w'x'y'} + \underset{12,13}{wxy'}$

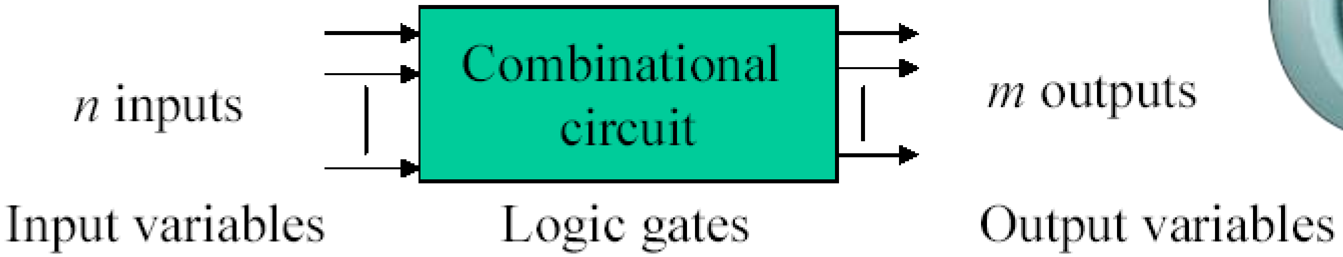
✓ ③ $F = yz + \underset{0,1}{w'x'y'} + \underset{8,12}{wy'z'} + \underset{13,15}{wxz}$

✓ ④ $F = yz + \overset{0\ \ 1}{w'x'y} + \overset{8,12}{wy'z'} + \overset{12,13}{wxy'}$

Combinational Circuits

- Logic circuits for digital system may be combinational or sequential
  - A combinational circuit consists of logic gates whose outputs are a function of the current inputs
  - Sequential circuits
    - contains storage elements in addition to logic gates
    - the outputs are a function of the current inputs and the state of the memory (storage) elements
    - The state of storage elements, in turn, is a function of previous inputs so outputs also depend on past inputs
    - They have feedback connection





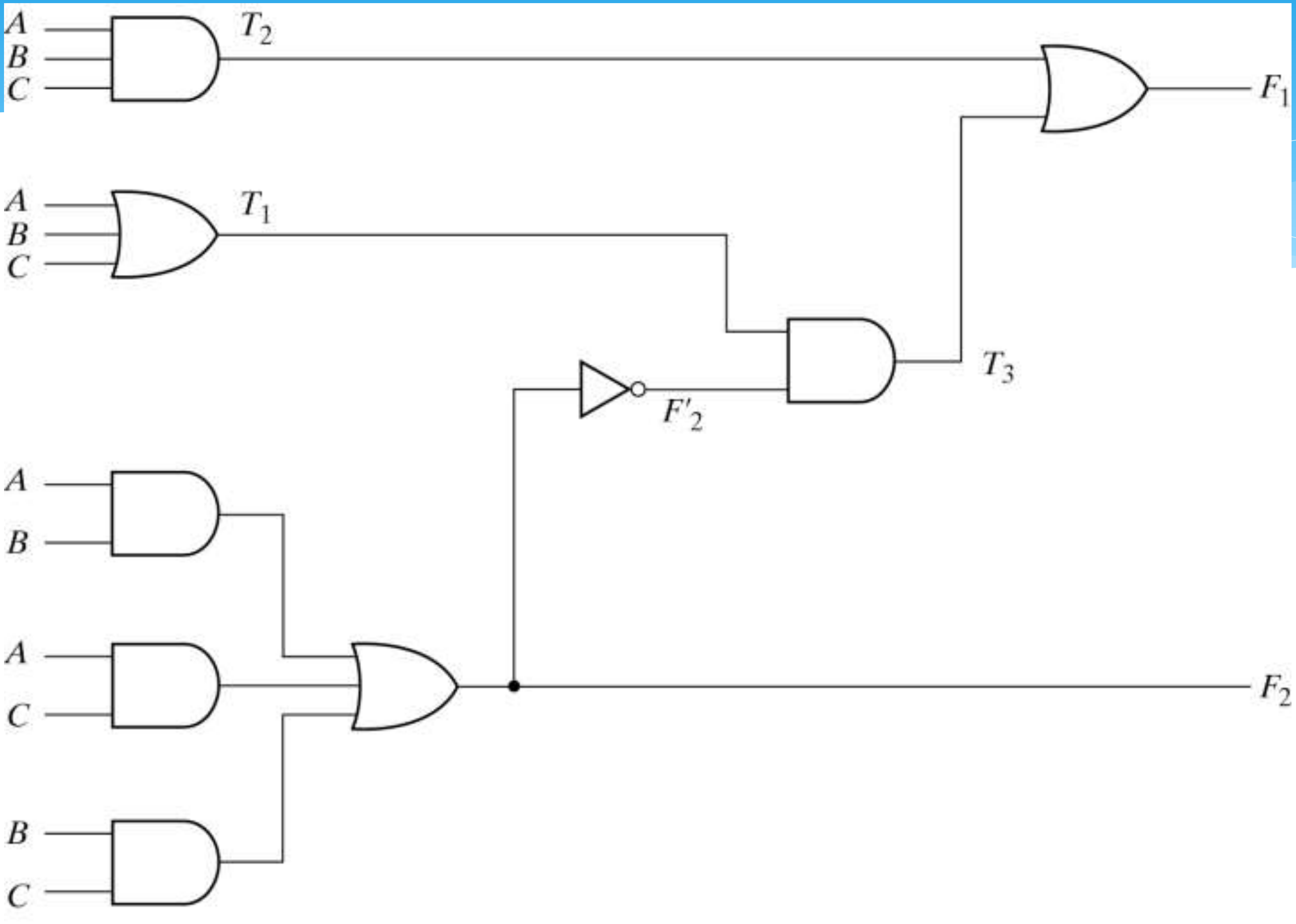
Analysis Procedure

- The analysis of a combinational circuit requires that we determine function that the circuit implements
- The first step in analysis procedure is to make sure that the given circuit is combinational and not sequential (No feedback path)
- To obtain the output Boolean function from a logic diagram we proceed as follows
  - Label all gate outputs that are a function of input variables with arbitrary symbols. Determine the Boolean function for each gate output
  - Label the gates that are a function of input variables and previously labeled gates with other arbitrary symbols. Find the Boolean function of these gates
  - Repeat the above process (step 2) until the outputs of the circuit are obtained
  - By repeated substitution of previously defined functions, obtain the output Boolean functions in terms of input variables

Combinational Circuits

- For n input variables there are 2^n possible binary input combinations
- For each possible input combination there is one possible output value
- The most important standard combinational circuits (discussed in this chapter) are adders, subtractors, comparators, decoders, encoders, and multiplexers
- If the storage registers are included with the combinational gates then this circuit is considered as sequential circuit

Analysis Procedure-Example



Analysis Procedure-Example

- The circuit has three binary inputs- A, B and C and two binary outputs- F_1 and F_2
- The outputs of various gates are labeled with intermediate symbols
- The output of gates that are a function of input variables are T_1 and T_2
- Output F_2 can be easily derived from the input variables
- The Boolean function for these three outputs are:
  - $F_2 = AB + AC + BC$
  - $T_1 = A + B + C$
  - $T_2 = ABC$
- Next we consider output of gates that are a function of already defined symbols
  - $T_3 = F'_2T_1$
  - $F_1 = T_3+T_2$

Analysis Procedure-Example

- To obtain F_1 as a function of A,B and C, from a series of substitutions proceed as follows

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

$$F_2 = AB + AC + BC$$

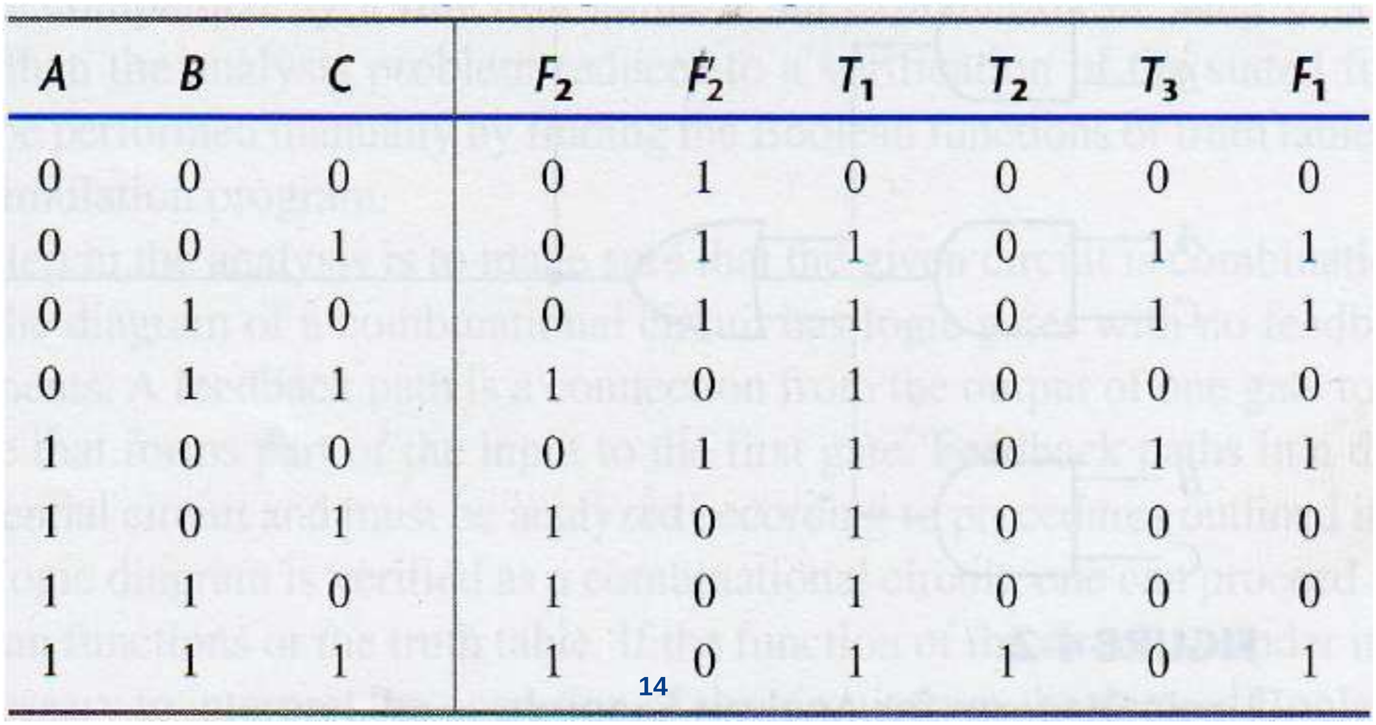
- These functions F_1 and F_2 implement circuit of a full-adder. Where:
  - F_1 is the sum
  - F_2 is the carry

Derivation of the Truth Table

- The derivation of the truth table for the circuit is a straight forward process once the output Boolean functions are known
- To obtain the truth table directly from the logic diagram without going through the derivations of the Boolean functions, we proceed as follows:
  - Determine the number of input variables in the circuit. For n inputs, form the 2^n possible input combinations and list binary numbers from 0 to $2^n – 1$ in a table
  - Label the outputs of selected gates with arbitrary symbols
  - Obtain the truth table for the outputs of those gates that are a function of the input variables only
  - Proceed to obtain the truth table for the outputs of those gates that are function of previously defined values until the columns for all outputs are determined

Truth Table for Fig 4-2

- F_2 equal to 1 for any combination that has 2 or 3 inputs equal to 1
- F_2’ is the complement of F2
- T_1 and T_2 are the OR and AND functions of input variables respectively
- T_3 = 1, when T_1 and F_2’ are equal to 1
- F_1 = 1, when either T_2 or T_3 or both are equal to 1



A	B	C	F_2	F_2'	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	1	0	1

Design Procedure

- The design procedure of combinational circuits involves following steps
  - State the problem (Circuit specifications)
  - From the circuit specifications determine the inputs and outputs
  - The input and output variables are assigned symbols
  - Derive the truth table that gives the relationship between inputs and outputs
  - Derive the simplified Boolean functions (simplify by algebraic manipulation or K-map method) for each output as a function of input variables
  - Draw the logic diagram and verify the correctness of the design

Practical Design Considerations

- Logic minimization must consider practical design constraints such as:
  - number of gates
  - number of inputs to a gate
  - propagation delay of signal through the gates
  - number of interconnection
  - limitations of the driving capabilities of each gate
- In most cases the simplification begins by satisfying an elementary objective – producing the simplified Boolean function in a standard form, and the proceed with further steps to meet other performance criteria

Code conversion example

- Different digital systems use different types of codes
- It is sometimes necessary to use the output of one system as the input to another
- A conversion circuit must be inserted between the two systems if each uses different codes for the same information
- A code converter is a circuit that makes the two systems compatible even though each uses a different binary code
- To convert from binary code A to binary code B, the input lines (of combinational circuit) must supply the bit combination of elements as specified by code A and the output lines must generate the corresponding bit combination of code B

Code conversion: BCD to Excess-3 Code

- Each code uses four bits to represent a decimal digit, there must be four input variables and four output variables
- Designate the four input binary variables by the symbols A,B,C,D and the four output variables by w, x, y and z.
- The truth table relating the input and output variable is made (shown in next slide)
- Four binary variables have 16 bit combinations but only 10 are listed in the truth table as 6 bit combinations not listed are don't care conditions. These have no meaning in BCD

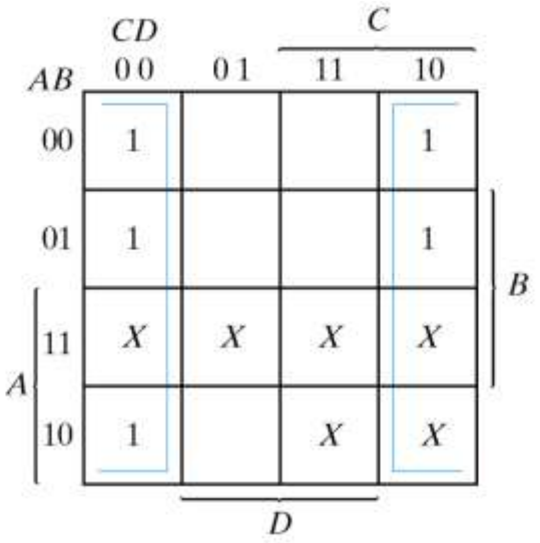
Code conversion:Truth Table

Input BCD				Output Excess-3 Code			
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

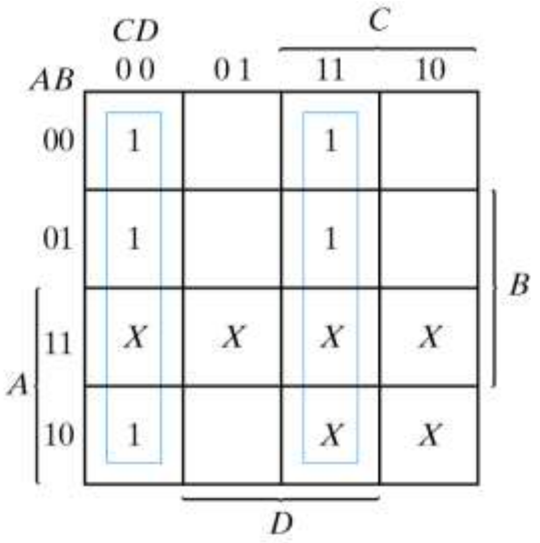
K-Maps

- There are four outputs, each as a function of four input variables.
- There are four maps, each representing one of the outputs, to obtain simplified Boolean functions.
- The 1's marked inside the squares are obtained from the minterms that make the output equal to 1
- The 1's are obtained from the truth table by going over the output columns one at a time.
- The 6 don't care minterms 10 through 15 are marked with an X

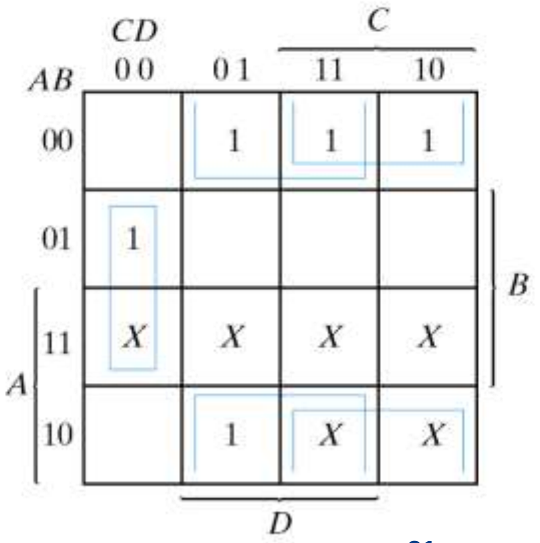
The Maps



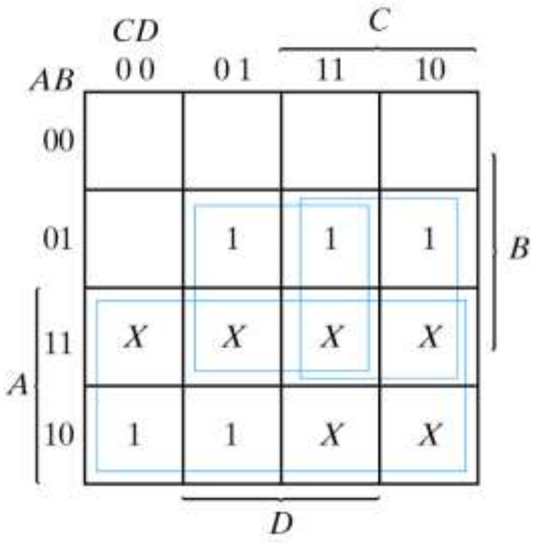
$z = D'$



$y = CD + C'D'$



$X = B'C + B'D + BC'D'$



$w = A + BC + BD$

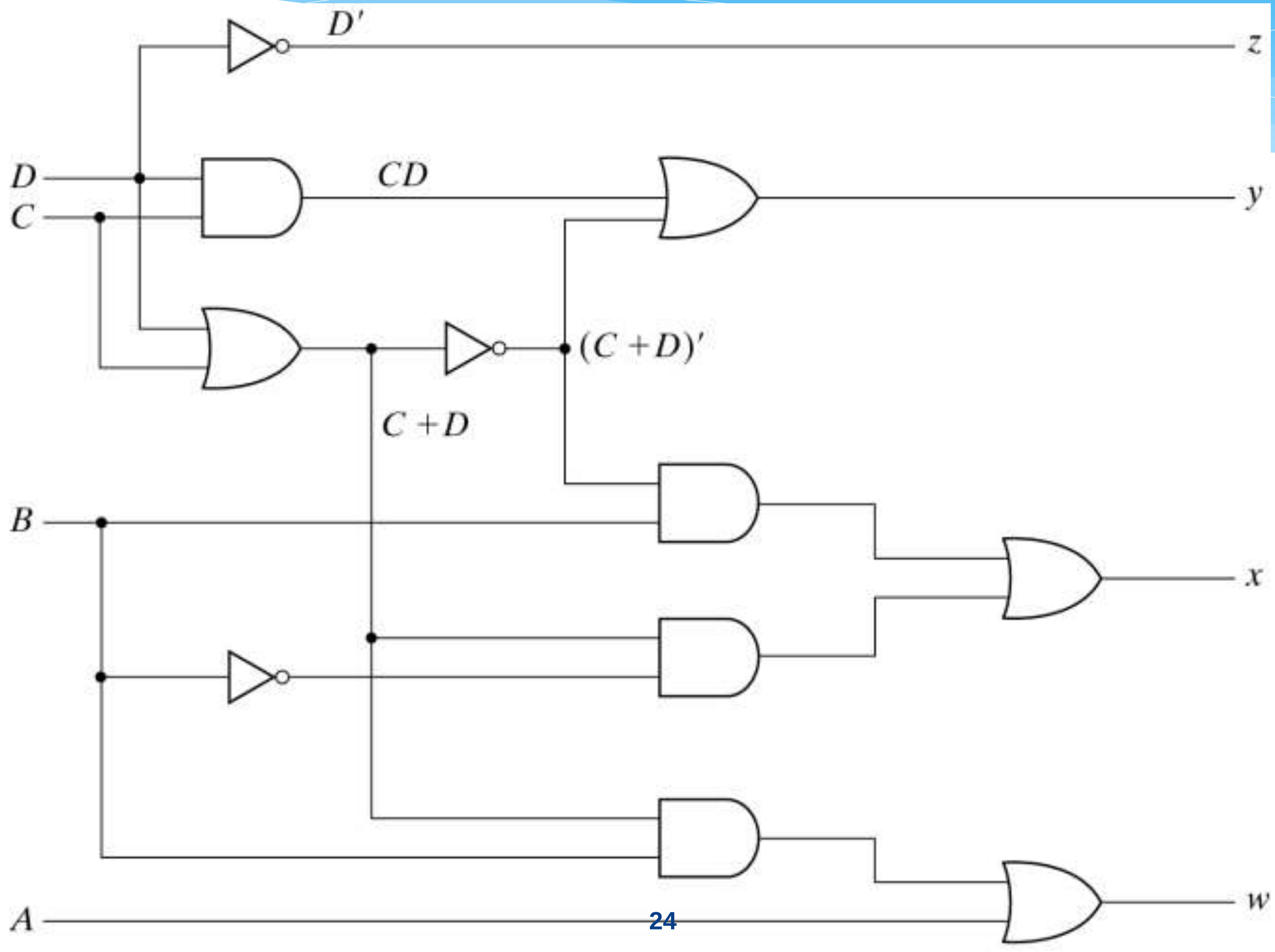
BCD to Excess-3 Code Converter

- The simplified functions
  - z = D'
  - y = CD +C'D‘
  - x = B'C + B'D+BC'D'
  - w = A+BC+BD
- A two-level logic diagram may be obtained directly from the above Boolean expressions but there are various other possibilities for logic diagram that implements this circuit
- Another implementation
  - z = D'
  - y = CD +C'D' = CD + (C+D)'
  - x = B'C + B'D+BC'D' = B'(C+D) +B(C+D)'
  - w = A+BC+BD

BCD-to-Excess Code Converter: continued

- The purpose this manipulation is to use common gates for two or more outputs
- This is implemented with three levels of gates
- OR gate with output C+D is used to implement partially each of three outputs
- Implementation with original sum of products require seven AND gates and three OR gates whereas after manipulation we require four AND gates and four OR gates
- In this way the logical circuit has been implemented in a much economical way.

Logic Diagram : BCD-to-Excess Code Converter



The End