Lecture (06)
Minterm, Maxterm (II)

By:

Dr. Ahmed ElShafee

Revision

- To find the expression of truth table,
  - Sum of minterms = 1 [1.1.1 + 1.1.1 + …..]
    - minterm = product of elements equals 1 with complement of elements equals 0
  - Product of maxterms = 0 [(0+0+0) . (0+0+0) . ….]
    - Maxterms = sun of elemets equals 0 with complement of elements equals 1

- *Functions of Three Variables*

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

$$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

$$f_1 = (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z)$$
$$= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$$

$$f_2 = (x + y + z)(x + y + z')(x + y' + z)(x' + y + z)$$
$$= M_0 M_1 M_2 M_4$$

General Minterm and Maxterm Expansions

- Following table represents a truth table for a general function of three variables.
- Because each *ai* can be specified in two ways, there are 2^8 ways of filling the *F* column of the truth table, therefore, there are 256 different functions of three variables.
- minterm expansion for a general function of three variables (if *ai* 0, the corresponding minterm is not present.)

A B C	F
0 0 0	a_0
0 0 1	a_1
0 1 0	a_2
0 1 1	a_3
1 0 0	a_4
1 0 1	a_5
1 1 0	a_6
1 1 1	a_7

$$F = a_0 m_0 + a_1 m_1 + a_2 m_2 + \cdots + a_7 m_7 = \sum_{i=0}^{7} a_i m_i$$

- The maxterm expansion for a general function of three variables is (if *ai* 1, *ai* *Mi* 1, and *Mi* drops out of the expansion)

$$F = (a_0 + M_0)(a_1 + M_1)(a_2 + M_2) \cdots (a_7 + M_7) = \prod_{i=0}^{7} (a_i + M_i)$$

- Domorgan's

$$F' = \left[\prod_{i=0}^{7}(a_i + M_i)\right]' = \sum_{i=0}^{7} a_i' M_i' = \sum_{i=0}^{7} a_i' m_i$$

$$F' = \left[\sum_{i=0}^{7} a_i m_i\right]' = \prod_{i=0}^{7}(a_i' + m_i') = \prod_{i=0}^{7}(a_i' + M_i)$$

- Generalizing Equations

$$F = \sum_{i=0}^{2^n-1} a_i m_i = \prod_{i=0}^{2^n-1}(a_i + M_i)$$

$$F' = \sum_{i=0}^{2^n-1} a_i' m_i = \prod_{i=0}^{2^n-1}(a_i' + M_i)$$

Mintem of two multiplied functions

- Given minterm expansions for two functions

$$f_1 = \sum_{i=0}^{2^n-1} a_i m_i \qquad f_2 = \sum_{j=0}^{2^n-1} b_j m_j$$

$$f_1 f_2 = \left(\sum_{i=0}^{2^n-1} a_i m_i\right)\left(\sum_{j=0}^{2^n-1} b_j m_j\right) = \sum_{i=0}^{2^n-1}\sum_{j=0}^{2^n-1} a_i b_j m_i m_j$$

$$= \sum_{i=0}^{2^n-1} a_i b_i m_i \quad (\text{because } m_i m_j = 0 \text{ unless } i = j)$$

- example

$$f_1 = \Sigma\, m(0, 2, 3, 5, 9, 11) \quad \text{and} \quad f_2 = \Sigma\, m(0, 3, 9, 11, 13, 14)$$

$$f_1 f_2 = \Sigma\, m(0, 3, 9, 11)$$

- Table summarizes the procedures for conversion between minterm and maxterm expansions of *F* and *F*,

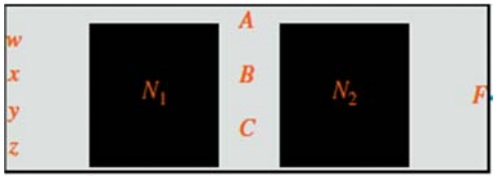
GIVEN FORM \ DESIRED FORM	Minterm Expansion of *F*	Maxterm Expansion of *F*	Minterm Expansion of *F′*	Maxterm Expansion of *F′*
Minterm Expansion of *F*	_____	maxterm are those not on the minterm list for *F*	list minterms not present in *F*	maxterm are the same as minterm of *F*
Maxterm Expansion of *F*	minterm are those not on the maxterm list for *F*	_____	minterm are the same as maxterm of *F*	list maxterms not present in *F*

- Example

GIVEN FORM \ DESIRED FORM	Minterm Expansion of *f*	Maxterm Expansion of *f*	Minterm Expansion of *f′*	Maxterm Expansion of *f′*
$f = \Sigma\, m(3, 4, 5, 6, 7)$	_____	$\Pi\, M(0, 1, 2)$	$\Sigma\, m(0, 1, 2)$	$\Pi\, M(3, 4, 5, 6, 7)$
$f = \Pi\, M(0, 1, 2)$	$\Sigma\, m(3, 4, 5, 6, 7)$	_____	$\Sigma\, m(0, 1, 2)$	$\Pi\, M(3, 4, 5, 6, 7)$

Incompletely Specified Functions

- A large digital system is usually divided into many subcircuits.
- Consider the following example in which the output of circuit *N*1 drives the input of circuit *N*2.



- there are no combinations of values for *w*, *x*, *y*, and *z* which cause *A*, *B*, and *C* to assume values of 001 or 110
- when we design *N*2, it is not necessary to specify values of *F* for *ABC* 001 or 110 because these combinations of values can never occur as inputs to *N*2

- The X's in the table indicate that we don't care whether the value of 0 or 1 is assigned to *F* for the combinations *ABC* 001 or 110
- The minterms *A'B'C* and *ABC'* are referred to as don't-care minterms,

A B C	F
0 0 0	1
0 0 1	X
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	0
1 1 0	X
1 1 1	1

Expression simplification of doesn't care terms

- When we realize the function, we must specify values for the don't-cares. It is will help to simplify the function.
- value 0 to both X's

$$F = A'B'C' + A'BC + ABC = A'B'C' + BC$$

- assign 1 to the first X and 0 to the second

$$F = A'B'C' + A'B'C + A'BC + ABC = A'B' + BC$$

- assign 1 to both X's,

$$F = A'B'C' + A'B'C + A'BC + ABC' + ABC = A'B' + BC + AB$$

- The second choice of values leads to the simplest solution.

A B C	F
0 0 0	1
0 0 1	X
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	0
1 1 0	X
1 1 1	1

Minterm expression of doesn't care terms

- When writing the minterm expansion for an incompletely specified function, we
- will use *m* to denote the required minterms and *d* to denote the don't-care minterms.

$$F = \Sigma\, m(0, 3, 7) + \Sigma d(1, 6)$$

A B C	F
0 0 0	1
0 0 1	X
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	0
1 1 0	X
1 1 1	1

Maxterm expression of doesn't care terms

$$F = \Pi\, M(2,4,5) \bullet \Pi\, D\,(1,6)$$

A B C	F
0 0 0	1
0 0 1	X
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	0
1 1 0	X
1 1 1	1

Example 01

- design a simple binary adder that adds two 1-bit binary numbers, *a* and *b*, to give a 2-bit sum. The numeric values for the adder inputs and output are as follows

a	b	Sum	
0	0	00	(0 + 0 = 0)
0	1	01	(0 + 1 = 1)
1	0	01	(1 + 0 = 1)
1	1	10	(1 + 1 = 2)

- *Input is A* and *B* and the 2-bit output sum by the logic variables *X* and *Y*,

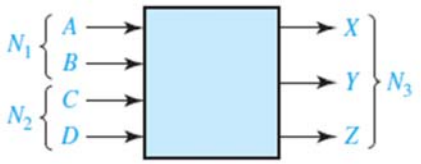
$$X = AB$$

$$Y = A'B + AB' = A \oplus B$$

A B	X Y
0 0	0 0
0 1	0 1
1 0	0 1
1 1	1 0

Example 2

- An adder is to be designed which adds two 2-bit binary numbers to give a 3-bit binary sum. Find the truth table for the circuit.



TRUTH TABLE:

N_1		N_2		N_3		
A	B	C	D	X	Y	Z
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

TRUTH TABLE:

N_1		N_2		N_3		
A	B	C	D	X	Y	Z
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

- Now we wish to derive the switching functions for the output variables. In doing so, we will treat *A*, *B*, *C*, *D*, *X*, *Y*, and *Z* as switching variables having nonnumeric values 0 and 1.
- (Remember that in this case the 0 and 1 may represent low and high voltages, open and closed switches, etc.)

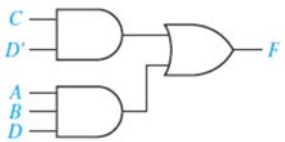
$$X(A, B, C, D) = \Sigma\, m(7, 10, 11, 13, 14, 15)$$
$$Y(A, B, C, D) = \Sigma\, m(2, 3, 5, 6, 8, 9, 12, 15)$$
$$Z(A, B, C, D) = \Sigma\, m(1, 3, 4, 6, 9, 11, 12, 14)$$

Example 03

- Design an error detector for 6-3-1-1 binary-coded-decimal digits. The output (*F*) is to be 1 if the four inputs (*A*, *B*, *C*, *D*) represent an invalid code combination.
- The valid 6-3-1-1 code combinations are listed in Table
- If any other combination occurs, this is not a valid 6-3-1-1 binary-coded-decimal digit, and the circuit output should be *F* 1

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

$$F = \Sigma\, m(2, 6, 10, 13, 14, 15)$$
$$= A'B'CD' + A'BCD' + AB'CD' + ABCD' + ABC'D + ABCD$$
$$= A'CD' + ACD' + ABD = CD' + ABD$$



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

Example 04

- The four inputs to a circuit (*A*, *B*, *C*, *D*) represent an 8-4-2-1 binary-coded-decimal digit.
- Design the circuit so that the output (*Z*) is 1 if the decimal number represented by the inputs is exactly divisible by 3.
- Assume that only valid BCD digits occur as inputs.

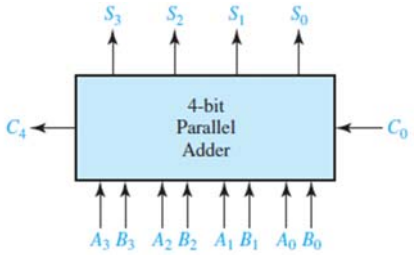
A	B	C	D	Z
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

- The digits 0, 3, 6, and 9 are exactly divisible by 3
- Z = 1 for the input combinations $ABCD$ 0000, 0011, 0110, and 1001
- The input combinations 1010, 1011, 1100, 1101, 1110, and 1111 do not represent valid *BCD* digits and will never occur

$$Z = \Sigma\, m(0, 3, 6, 9) + \Sigma\, d(10, 11, 12, 13, 14, 15)$$

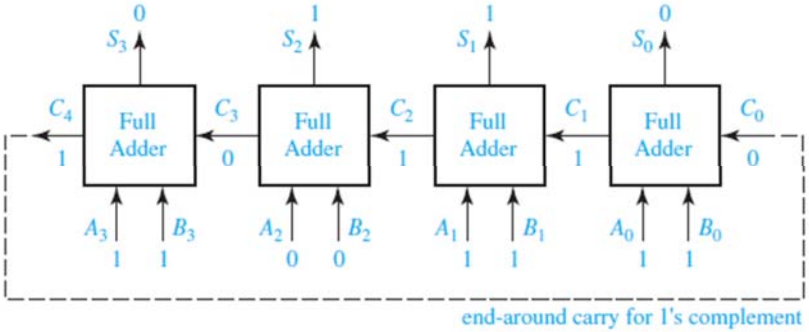
Design of Binary Adders and Subtracters

- we will design a parallel adder that adds two 4-bit unsigned binary numbers and a carry input to give a 4-bit sum and a carry output



- construct a truth table with nine inputs and five outputs and then derive and simplify the five output equations
- this approach would be very difficult, and the resulting logic circuit would be very complex.

- A better method is to design a logic module that adds two bits and a carry, and then connect four of these modules together to form a 4-bit adder



- 10110 (carries)
  1011
  + 1011
  10110
- $A0 + B0 + C0 = 1 + 1 + 0 = 10_2$ → $S0 = 0$ and $C1 = 1$
- $A1 + B1 + C1 = 1 + 1 + 1 = 11_2$ → $S1 = 1$ and $C2 = 1$.

- Figure gives the truth table for a full adder with inputs *X*, *Y*, and *Cin*.
- The outputs for each row (*X* +*Y* +*Cin*)
- splitting the result into a carry out (*Ci*1) and a sum bit (*Si*).
- Example:
- 101 row
- 1 +0 +1 = 10_2, so *Ci*1 =1 and *Si* = 0.

X	Y	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

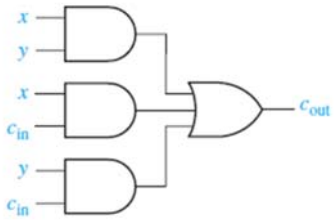
$$Sum = X'Y'C_{in} + X'YC'_{in} + XY'C'_{in} + XYC_{in}$$
$$= X'(Y'C_{in} + YC'_{in}) + X(Y'C'_{in} + YC_{in})$$
$$= X'(Y \oplus C_{in}) + X(Y \oplus C_{in})' = X \oplus Y \oplus C_{in}$$

X	Y	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

-

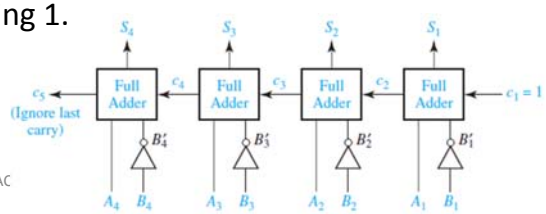
X	Y	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

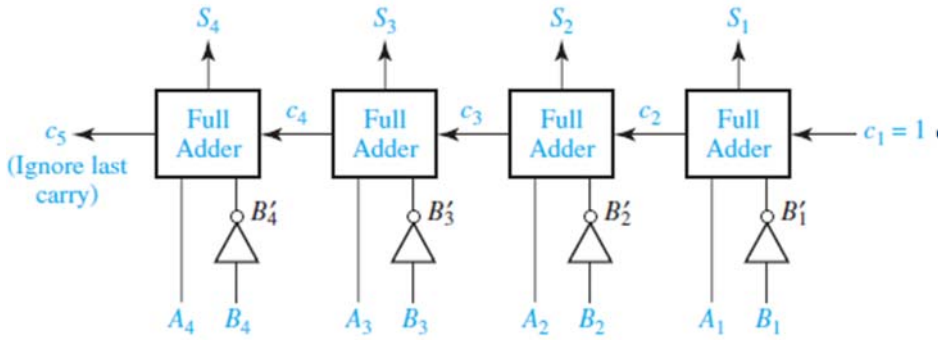
$$C_{out} = X'YC_{in} + XY'C_{in} + XYC'_{in} + XYC_{in}$$
$$= (X'YC_{in} + XYC_{in}) + (XY'C_{in} + XYC_{in}) + (XYC'_{in} + XYC_{in})$$
$$= YC_{in} + XC_{in} + XY$$



Subtracter using full adder

- Subtraction of binary numbers is most easily accomplished by adding the complement of the number to be subtracted,
- To compute *A* - *B*, add the complement of *B* to *A*. This gives the correct answer because *A* + (-*B*) =*A* -*B*.
- Either 1's or 2's complement is used depending on the type of adder employed.
- The 2's complement of *B* can be formed by first finding the 1's complement and then adding 1.





$A = 0110$ (+6)
$B = 0011$ (+3)
The adder output is

```
  0110     (+6)
 +1100     (1's complement of 3)
 +   1     (first carry input)
(1) 0011 = 3 = 6 − 3
```


Minimum Forms of Switching Function

- The Karnaugh map techniques developed in this unit lead directly to minimum cost two-level circuits composed of AND and OR gates.
- An expression consisting of a sum of product terms corresponds directly to a two-level circuit composed of a group of AND gates feeding a single OR gate.
- Similarly, a product-of-sums expression corresponds to a two-level circuit composed of OR gates feeding a single AND gate

minimum sum-of-products expression:

(a) has a minimum number of terms and

(b) has a minimum number of literals.

- To obtain minimum sum-of-products from minterm procedure:
- **1.** Combine terms by using XY'+XY=X. Do this repeatedly to eliminate as many literals as possible. A given term may be used more than once because (X+X=X).
- **2.** Eliminate redundant terms

Example 05

- Find a minimum sum-of-products expression for

$$F(a, b, c) = \Sigma m\,(0, 1, 2, 5, 6, 7)$$

- Combine terms by using XY'+XY=X. Do this repeatedly to eliminate as many literals as possible. A given term may be used more than once because (X+X=X).

$$F = a'b'c' + a'b'c + a'bc' + ab'c + abc' + abc$$
$$= a'b' + b'c + bc' + ab$$

- Eliminate redundant terms

$$F = a'b'c' + a'b'c + a'bc' + ab'c + abc' + abc$$
$$= a'b' + bc' + ac$$

minimum product-of-sums expression:

- (a) has a minimum number of factors, and
- (b) has a minimum number of literals.

Procedures

- the theorem (X+Y).(X+Y')=X is used to combine terms.

Example 06

- $F = \prod M(10, 8, 9, 1, 13, 6)$

$$(A + B' + C + D')(A + B' + C' + D')(A + B' + C' + D)(A' + B' + C' + D)(A + B + C' + D)(A' + B + C' + D)$$
$$= (A + B' + D') \quad (A + B' + C') \quad (B' + C' + D) \quad (B + C' + D)$$
$$= (A + B' + D') \quad (A + B' + C') \quad (C' + D)$$


Thanks,..
See you next week (ISA),…
٢٧
Dr. Ahmed ElShafee, ACU : Spring 2016, Logic Design