King Fahd University of Petroleum & Minerals Computer Engineering Dept

COE 202 – Fundamentals of Computer Engineering

Term 062

Dr. Ashraf S. Hasan Mahmoud

Rm 22-148-3

Ext. 1724

Email: ashraf@ccse.kfupm.edu.sa

Karnaugh Map (K-Map)

- A tabular method to simplify function expressions – an alternative to algebraic manipulation
- Produces 2-level (sum of products or product of sums) implementation

1-variable K-map

- Consider the function F(X)

F(X) = 0
(F is NOT dependent on X)

x	0	1
	0	0

F(X) = 1
(F is NOT dependent on X)

x	0	1
	1	1

F(X) = X

x	0	1
	0	1

F(X) = X'

x	0	1
	1	0

2-variable K-map

- Consider the function F(X,Y)
- The general 2-variable K-map is as shown
- The map is formed by putting two 1-variable K-maps side by side

Y=1

X \ Y	0	1
0	X'Y' (m_0)	X'Y (m_1)
X=1 — 1	XY' (m_2)	XY (m_3)

- Examples:

X \ Y	0	1
0	0	0
1	0	1

F(X,Y) = XY

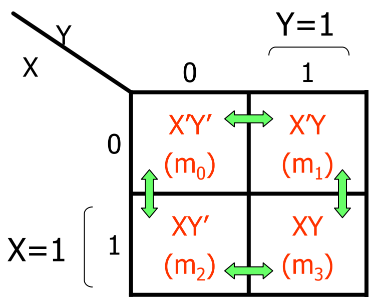
X \ Y	0	1
0	0	1
1	1	1

F(X,Y) = X'Y+XY'+XY
= X +Y

2-variable K-map – cont'd

- Neighbors sharing one literal:
  - X'Y' (or m_0) and X'Y (or m_1) ➔ sharing the literal X'
  - X'Y' (or m_0) and XY' (or m_2) ➔ sharing the literal Y'
  - X'Y (or m_1) and XY (or m_3) ➔ sharing the literal Y
  - XY' (or m_2) and XY (or m_3) ➔ sharing the literal X
- If for example

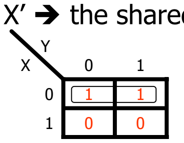
  F(X,Y) = m0 + m1 = X'Y' + X'Y



Then once can simplify F as follows:

F(X,Y) = X'(Y' + Y)

= X' ➔ the shared literal



2-variable K-map – cont'd

- Example2: F(X,Y) = Σm(1,2,3)

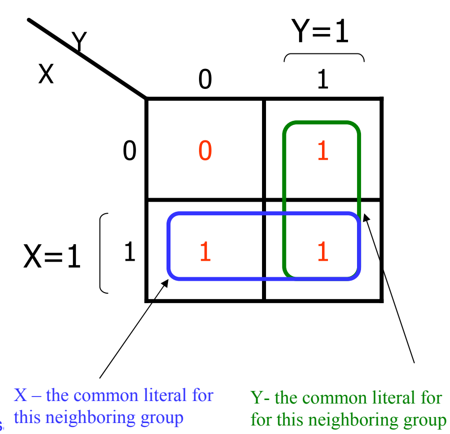
F can be simplified as in

F = X'Y +XY' + XY

= X'Y + XY + XY' + XY

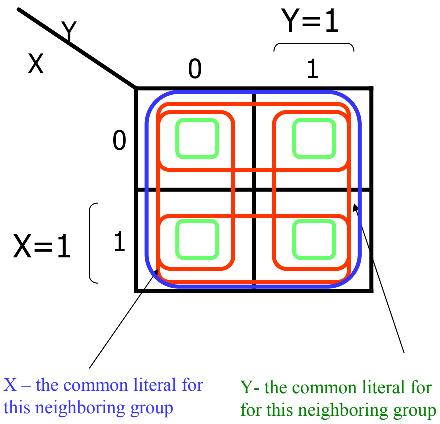
= (X'+X)Y + X(Y'+Y)

= Y + X



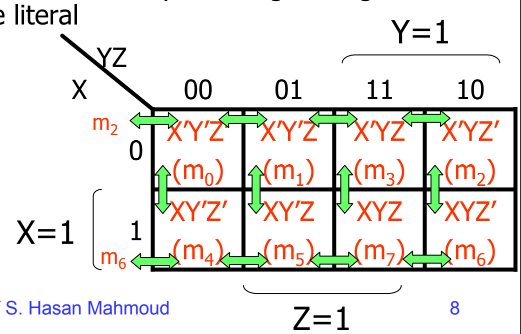
2-variable K-map – All Possible Squares

- 4 Groups each of one minterm
- 4 groups each of two minterms
- 1 group of 4 minterms



3-variable K-map

- Consider the function F(X,Y,Z)
- The general 3-variable K-map is as shown
- The map is formed by putting two 2-variable K-maps side by side
- Note:
  - The minterms are ordered such that any two neighboring minterm differ only in one literal
  - The K-Map (the numbering of the minterms) assumes X is the most significant variable and Z is the least significant variable



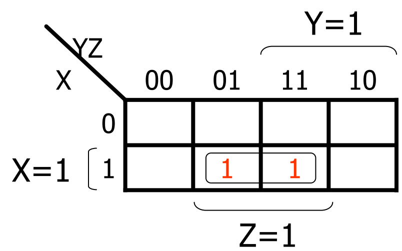
3-variable K-map – cont'd

- Example $F(X,Y,Z) = m_5 + m_7$

Once can simplify as in

$F = m_5 + m_7$
$= XY'Z + XYZ$
$= XZ(Y'+Y)$
$= XZ$

Or once can use the K-map as shown



The common literals for this group is XZ (they differ in Y)

Therefore: F(X,Y) = XZ

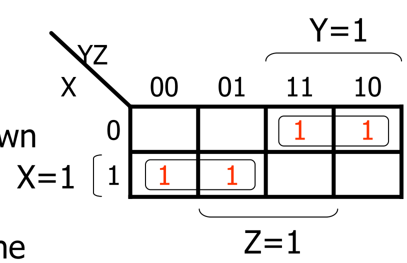
3-variable K-map – cont'd

- Example $F(X,Y,Z) = \Sigma m(2,3,4,5)$

Once can simplify as in

$F = m_2 + m_3 + m_4 + m_5$
$= X'YZ' + X'YZ + XY'Z' + XY'Z$
$= X'Y(Z'+Y) + XY'(Z'+Z)$
$= X'Y + XY'$

Or one can use the K-map as shown



The common literals for the 1st group is X'Y (they differ in Z), while the common literals for the 2nd group is XY' (they differ in Z)

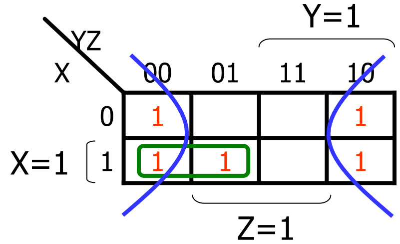
Therefore: F(X,Y) = X'Y + XY'

3-variable K-map – cont'd

- Example $F(X,Y,Z) = \Sigma m(0,2,4,5,6)$

Once can simplify as in

$F = m_0 + m_2 + m_4 + m_5 + m_6$
$= X'Y'Z' + X'YZ' + XY'Z' + XY'Z + XYZ'$
$= X'Y'Z' + X'YZ' + XY'Z' + XY'Z' + XY'Z + XYZ'$
$= Y'Z'(X' + X) + YZ'(X'+X) + XY'(Z'+Z')$
$= Z'(Y'+Y) + XY'$
$= Z' + XY'$



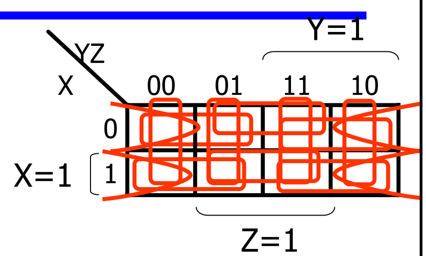
Or one can use the K-map as shown

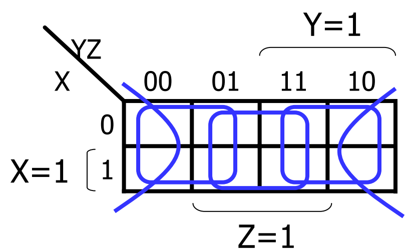
The common literals for the 1st group is XY' (they differ in Z), while the common literal for the 2nd group is Z' (they differ in XY)

Therefore: $F(X,Y) = Z' + XY'$

3-variable K-map – All Possible Groups

- 8 groups each of 1 minterms
- 12 groups each of 2 minterms
- 4 groups each of 4 minterms
- 1 group of 8 minterms



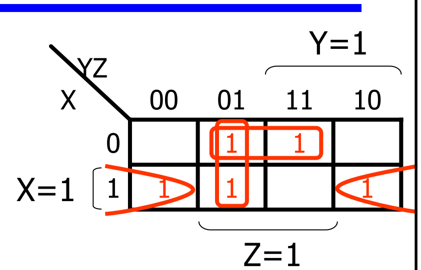


Rules for Choosing Groups

- The groups SHOULD cover all minterms
- The groups SHOULD have minimum overlap
- The groups SHOULD be maximized in size (to reduce their number or product terms)

Example

- Consider $F(X,Y,Z) = \Sigma m(1,3,4,5,6)$



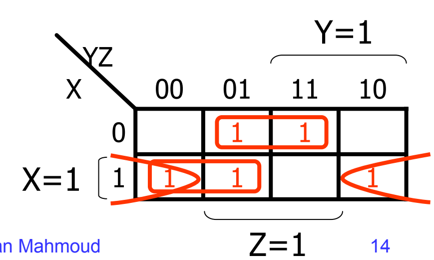
Following the groups selection rules:

- there is no group of 8 or 4 that can be selected
- there are only groups of 2 that can be selected
- once can select the groups as shown (minimum no of groups)

Therefore $F(X,Y,Z) = X'Z+Y'Z+XZ'$

OR (See second K-Map)

$F(X,Y,Z) = X'Z + XZ' + XY'$



The simplest expression is NOT unique!

Definitions

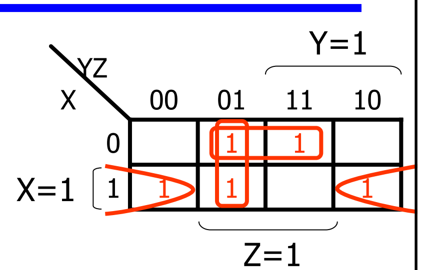
- Map manipulation: to minimize number of terms (i.e. simplify function and avoid redundant terms)
- Implicant:
  - A product term
  - Any valid square or group
- Prime Implicant: If you list all implicants, the removal of any literal does not lead to an implicant – the orignal implicant is a prime implicant
  - i.e. largest possible square
- Essential Prime Implicant: A prime implicant covering a minterm no other prime implicant does

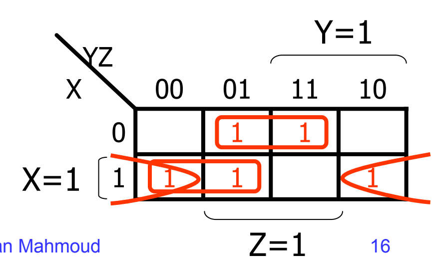
Example

- Consider F(X,Y,Z) = Σm(1,3,4,5,6)

List all implicants, prime implicants and essential prime implicants

- Solution:
- Implicants: XY′Z′, XZ′, XY′, XY′Z, X′Y′Z, Y′Z, …
- P.Is: XY′, XZ′, Y′Z, X′Z
- EPIs: X′Z, XZ′

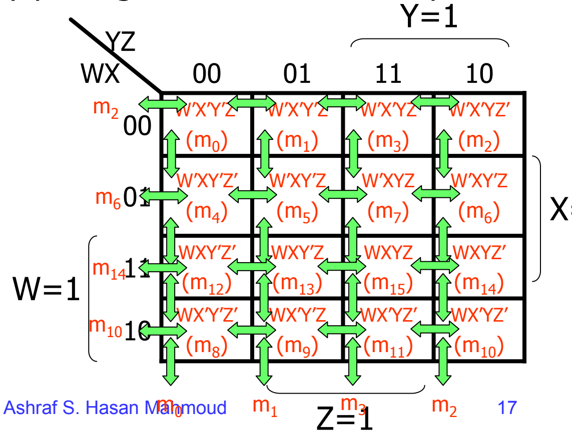




The simplest expression is NOT unique!

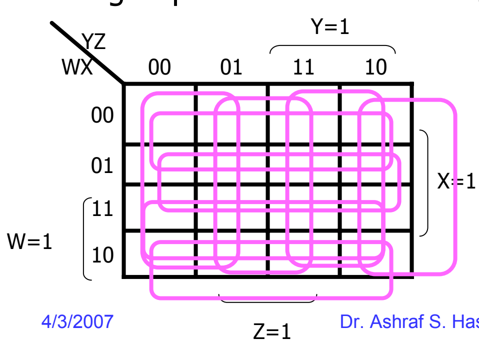
4-variable K-map

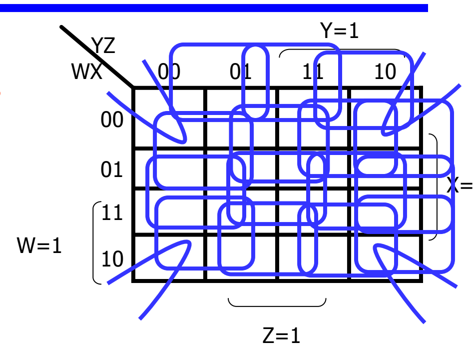
- Consider the function F(W,X,Y,Z)
- The general 4-variable K-map is as shown
- The map is formed by putting two 3-variable K-maps on top of each other
- Note:
  - The minterms are ordered such that any two neighboring minterm differ only in one literal
  - The K-Map (the numbering of the minterms) assumes W is the most significant variable and Z is the least significant variable



4-variable K-map – All Possible Groups

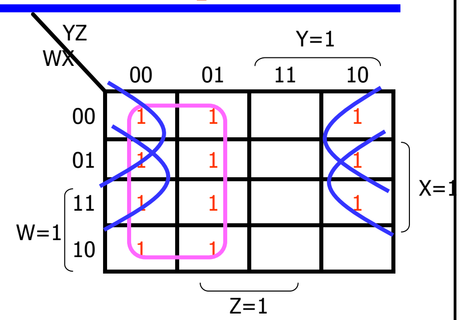
- 16 groups of one minterm
- 32 groups of two minterms
- ? groups of four minterms
- 8 groups of 8 minterms
- 1 group of 16 minterms





4-variable K-map – Example

- Consider F(W,X,Y,Z) = Σm(0,1,2,4,5,6,8,9,12,13,14)

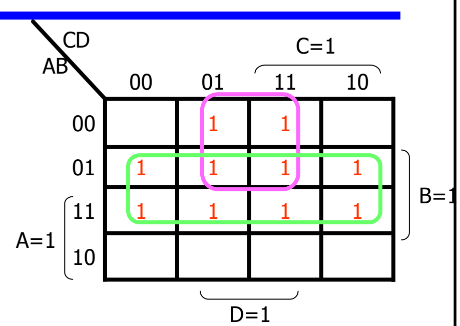


$F(W,X,Y,Z) = Y' + W'Z' + XZ'$

Example

- Consider F(A,B,C,D) = Σm(1,3,4,5,6,7,12,13,14, 15)

  Find essential prime implicants?



- Solution:

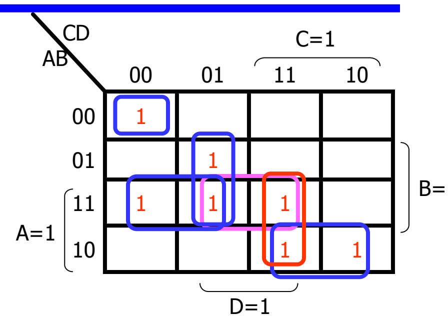
  B, A'D are EPI

What is F(A,B,C,D)?

Example

- Consider F(W,X,Y,Z) = Σm(0,1,5,10,11,12,13,15)

  Find all essential prime implicants? Write all possible expressions for F?



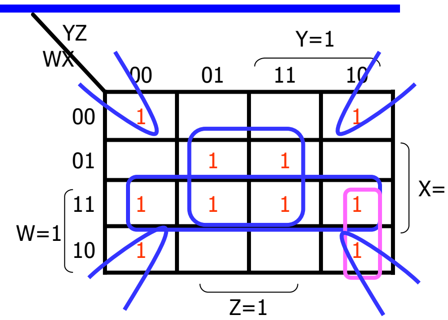
- Solution:

  EPI = A′B′C′D′, BC′D, ABC′, AB′C

  F(A,B,C,D) =
  A′B′C′D′ + BC′D + ABC′ + AB′C +
  ACD (or ABD)

Problem 2-19(a)

- Consider F(W,X,Y,Z) = Σm(0,2,5,7,8,10,12,13,14,15)
  Find all implicants, prime implicants, and essential prime implicants? Write all possible expressions for F?



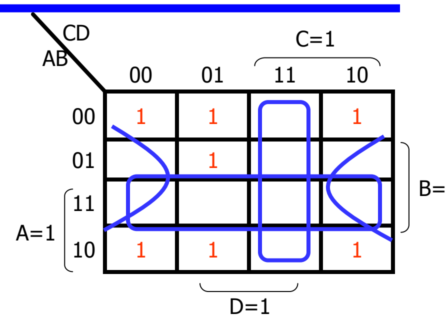
- Solution:

Implicants: W′X′Y′Z′, W′X′Z′, WX′Y′Z′, WX′YZ′, WX′Z′, X′Z′, …
PIs: WZ′, XZ, X′Z′, WX
EPIs: X′Z′ (only PI covering W′X′Y′Z′),
XZ (only PI covering W′XY′Z or W′XYZ)

F(W,X,Y,Z) = X′Z′ + XZ + WX, OR
= X′Z′ + XZ + WZ′

Product of Sums Simplification - Example

- Consider $F(A,B,C,D) = \Sigma m(0,1,2,5,8,9,10)$
  Write F in the simplified product of sums
- Solution:
  Follow same rule as before but for the ZEROs



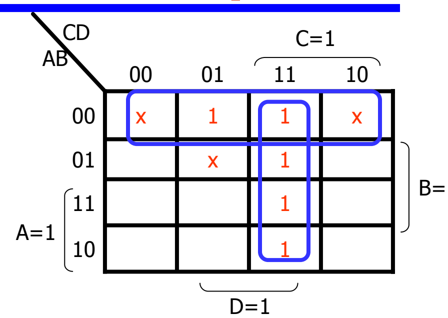
$F' = AB + CD + BD'$

Therefore,

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

Don't Care Conditions - Example

- Consider
  $F(A,B,C,D) = \Sigma m(1,3, 11, 15)$
  $d(A,B,C,D) = \Sigma m(0,2,5)$
  Simplify F.
- Solution:
  $F = A'B' + CD$

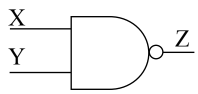


Note – for the simplification we assumed $A'B'C'D'$ (m_0) = 1, and $A'B$
CD' (m_2) = 1, while $A'BC'D$ (m_5) = 0

More Gates: NAND - NOR

X	Y	Z=(XY)'
0	0	1
0	1	1
1	0	1
1	1	0

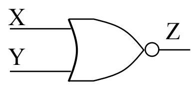
NAND



F = (XY)'

X	Y	Z=(X+Y)'
0	0	1
0	1	0
1	0	0
1	1	0

NOR

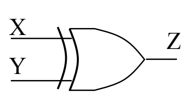


F = (X+Y)'

- Sometimes it is desirable to build circuits using NAND gates only or NOR gates only

More Gates: XOR - XNOR

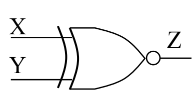
Exclusive OR (XOR)



F = X'Y + XY'
= X⊕Y

X	Y	Z=X⊕Y
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive NOR (XNOR)

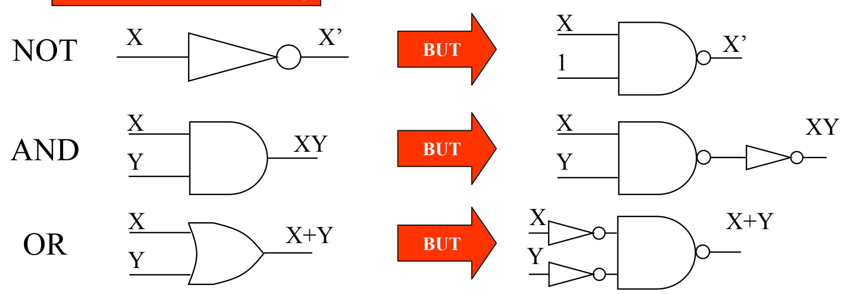


F = XY + X'Y'
= (X⊕Y)'

X	Y	Z=(X⊕Y)'
0	0	1
0	1	0
1	0	0
1	1	1

NAND Circuits

We have learned how to build any function using

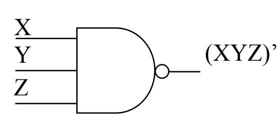


- Therefore, we can build all functions we learned so far using NAND gates ONLY
- NAND is a UNIVERSAL gate

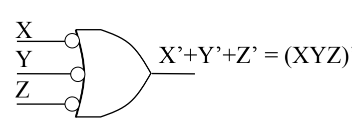
Graphic Symbols for NAND Gate

- Two equivalent graphic symbols or shapes for the SAME function

AND-NOT



NOT-OR

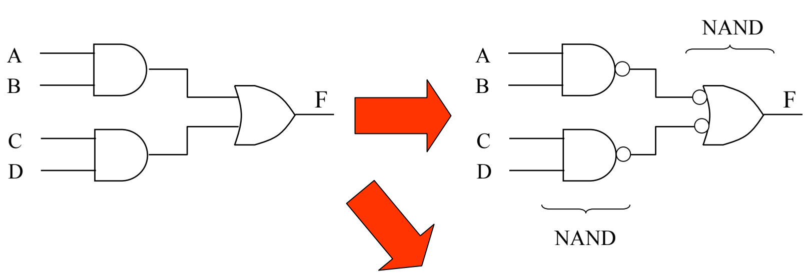


Two Level Implementation

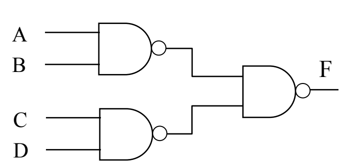
- We use the sum of products form
  - This results from K-map simplification or algebraic manipulation
- The AND gates – 1st level
- The OR gate – 2nd level
- Inverters to inputs of ANDs or output of OR are not counted as levels

Two Level Implementation – cont'd

- Example: Consider F = AB + CD

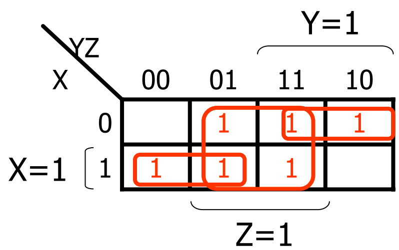


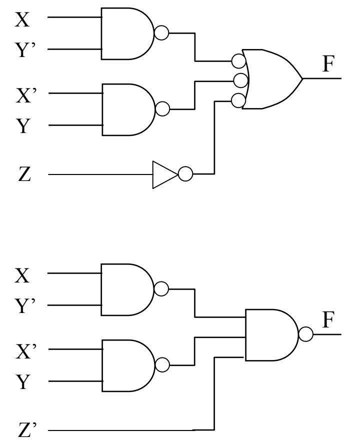

```
Proof:
    F = ((AB)'.(CD)')'
      = ((AB)')' + ((CD)')'
      = AB + CD
```



Example

- Consider F =Σm(1,2,3,4,5,7) – Implement using NAND gates





F(X,Y) = Z+XY′+X′Y

Rules for 2-Level NAND Implementations

- Simplify the function and express it in sum-of-products form
- Draw a NAND gate for each product term (with 2 literals or more)
- Draw a single NAND gate at the 2nd level (in place of the OR gate)
- A term with single literal requires a NOT

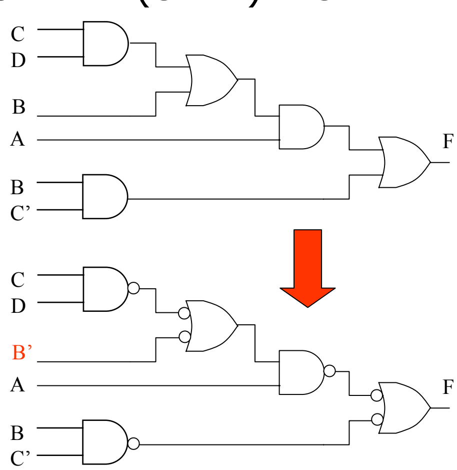
What about multi-level circuits?

Rules for Multi-Level NAND Implementations

- NOTE: the function is NOT in the standard form – WHY?
- Steps:
  - Draw a NAND gate for each AND gate
  - Draw a NAND gate (using the NOT-OR symbol) for each OR gate
  - Check paths – add inverters to make even number of bubbles

Example (1): Rules for Multi-Level NAND Implementations

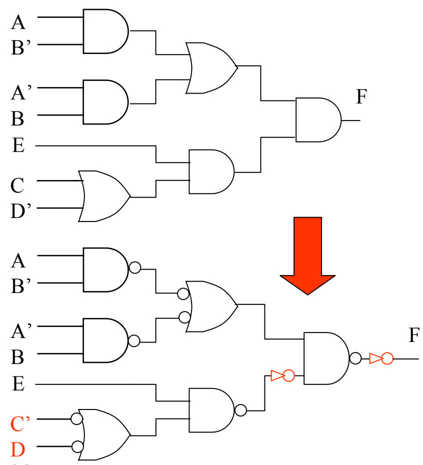
- Consider the function F = A(CD+B)+BC′



Note: an inverter is added to the path for the literal B

Example (2): Rules for Multi-Level NAND Implementations

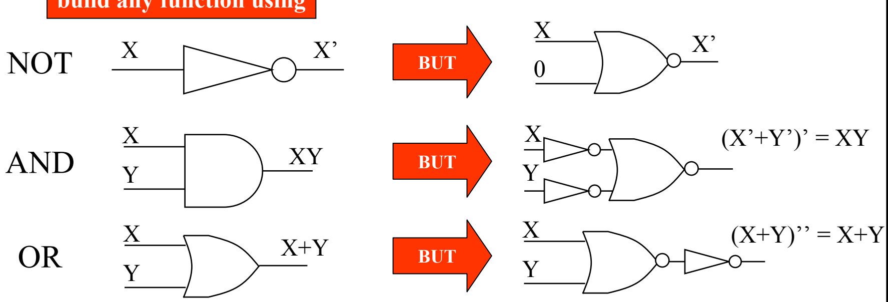
- Consider the function F = (AB′+A′B)E(C+D′)



Note: an inverter is added to three paths

NOR Circuits

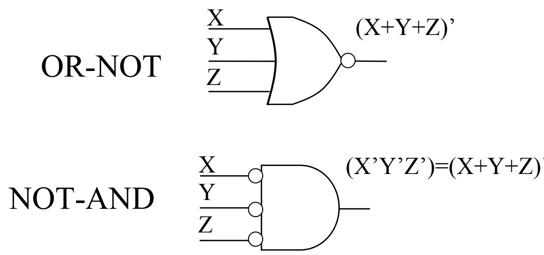
We have learned how to build any function using



- Therefore, we can build all functions we learned so far using NOR gates ONLY
- NOR is a UNIVERSAL gate

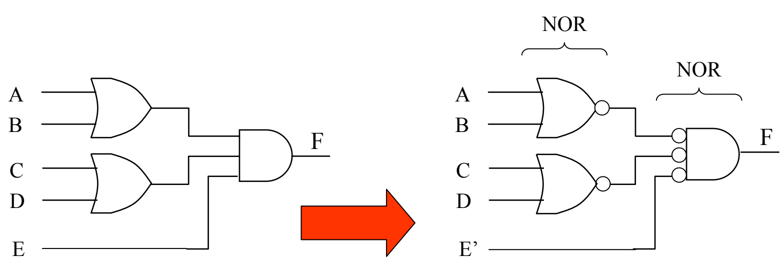
Graphic Symbols for NOR Gate

- Two equivalent graphic symbols or shapes for the SAME function



Two Level Implementation - NOR

- Consider F = (A+B)(C+D)E

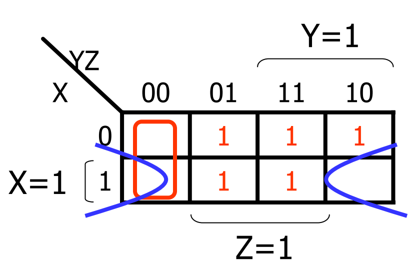


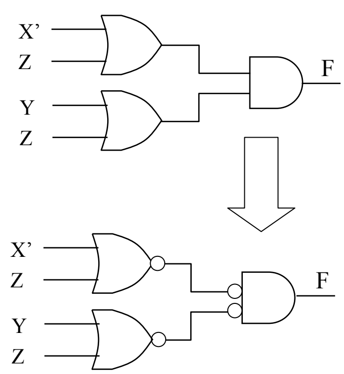
Two Level Implementation - NOR

- We use the *product of sums* form
  - This results from K-map simplification or algebraic manipulation
  - Note to get product of sums we the zeros – simplify and then complement the expression
- The OR gates – 1st level
- The AND gate – 2nd level
- Inverters to inputs of ORs or output of AND are not counted as levels

Example

- Consider F =Σm(1,2,3,5,7) – Implement using NOR gates





F'(X,Y) = Y'Z'+XZ', or
F(X,Y) = (Y+Z)(X'+Z)

Rules for 2-Level NOR Implementations

- Simplify the function and express it in product of sums form
- Draw a NOR gate (using OR-NOT symbol) for each sum term (with 2 literals or more)
- Draw a single NOR gate (using NOT-AND symbol) the 2nd level (in place of the AND gate)
- A term with single literal requires a NOT

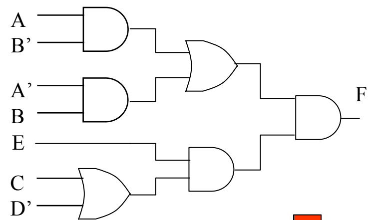
What about multi-level circuits?

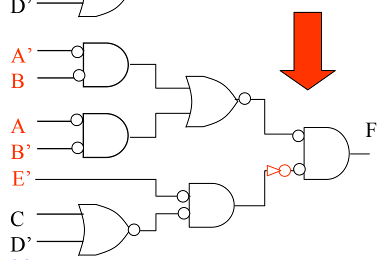
Rules for Multi-Level NOR Implementations

- NOTE: the function is NOT in the standard form – WHY?
- Steps:
  - Draw a NOR (OR-NOT) gate for each OR gate
  - Draw a NOR (NOT-AND) gate for each AND gate
  - Check paths – add inverters to make even number of bubbles

Example: Rules for Multi-Level NOR Implementations

- Consider the function F = (AB′+A′B)E(C+D′)

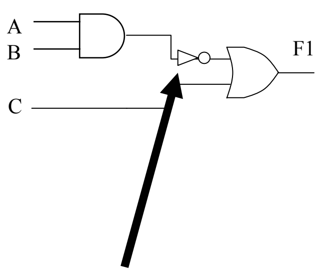


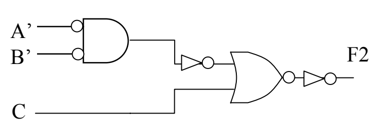


Note: an inverter is added to six paths

Example (2): Rules for Multi-Level NOR Implementations

- Consider the function F = (AB)′+C



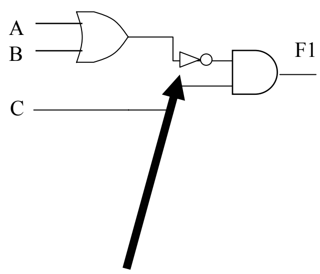


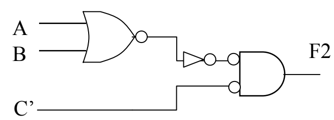
- Hence this bubble need not be complemented
- Only the ones added!!

F2 = ((A′′B′′)′+C)′′
= (AB)′+C
= F1

Example (3): Rules for Multi-Level NOR Implementations

- Consider the function F = (A+B)′C





- Hence this bubble need not be complemented
- Only the ones added!!

```
F2 = (A+B)'''C''
   = (A+B)'C'
   = (A+B)' C
   = F1
```


Example (4):

- **<u>Problem:</u>** Simplify the following expression and implement using 2-level circuit with NAND gates. Assume both true and complement version of the input variables are available.

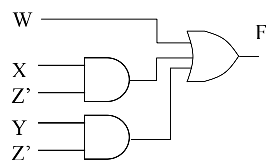
  WX′ + WXZ + W′Y′Z + W′XZ′ + WXZ′

Example (4): cont'd

- **Solution:**
- Simplified 2-level implementation → K-Map

  F(W,X,Y,Z) = WX' + WXZ + W'Y'Z' + W'XY' + WXZ'

F(W,X,Y,Z) = W + XZ' + YZ'

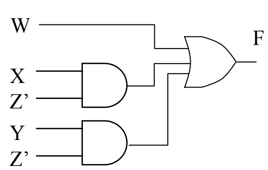


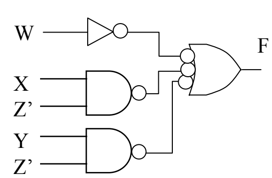
WX \ YZ	00	01	11	10
00	1			
01	1	1		
11	1	1	1	1
10	1	1	1	1

Example (4): cont'd

- **Solution:**

  F(W,X,Y,Z) = W + XZ' + YZ'





XOR-XNOR Gates and Parity Checking

- **Required Material**
- **Refer to lesson2_7.pdf at:**

http://www.ccse.kfupm.edu.sa/~elrabaa/coe202/Lessons/Lesson2_7.pdf