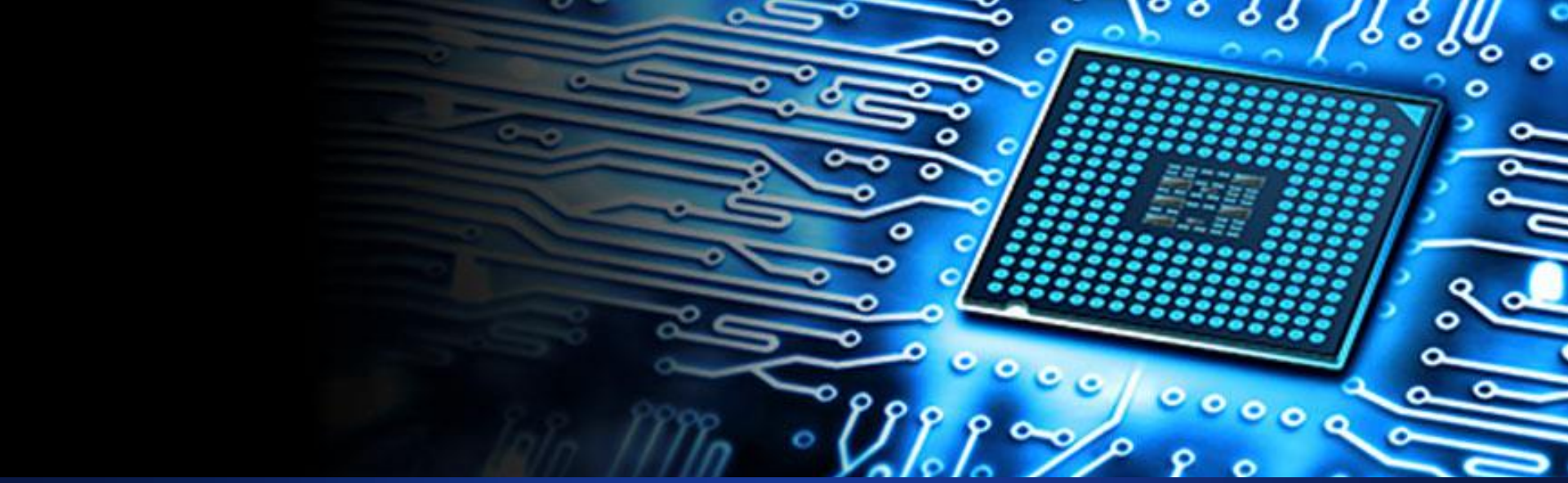


Chapter 1

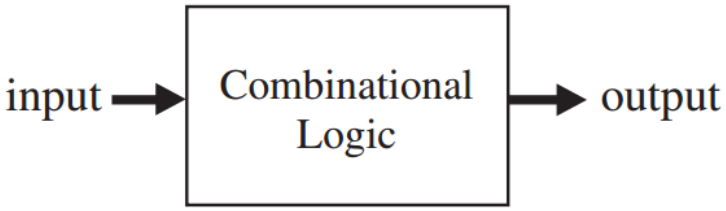
Review of Logic Design Fundamentals

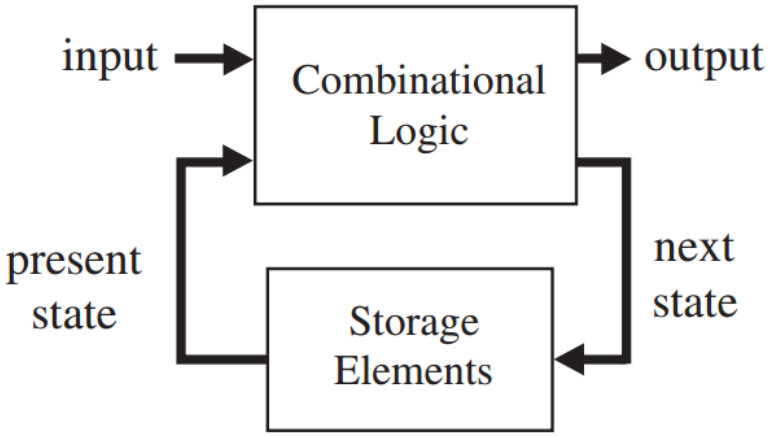
Version: 2023/11/14

	Contents
1.1	**Combinational Logic（组合逻辑）**
1.2	Boolean Algebra and Algebra Simplification
1.3	Karnaugh Maps
1.4	Designing with NAND and NOR Gates
1.5	Hazards in Combinational Circuits
1.6	Flip-Flops and Latches
1.7	Mealy Sequential Circuit Design
1.8	Moore Sequential Circuit Design
1.9	Equivalent States and Reduction of State Tables
1.10	Sequential Circuit Timing
1.11	Tristate Logic and Busses

1.1 Combinational Logic

	Memory	The present output depends ...
Combinational logic (组合逻辑)	X	only on the present input
Sequential logic (时序逻辑)	√	on the present input and **past sequence of inputs**





1.1 Combinational Logic

- Logic values: 0, 1
- Positive logic & Negative logic

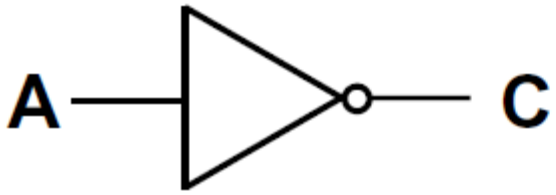
	logic 0	logic 1
Positive logic	low voltage	high voltage
Negative logic	high voltage	low voltage

1.1 Combinational Logic

Basic gates

AND: C = A B

OR: C = A + B



NOT: C = A'

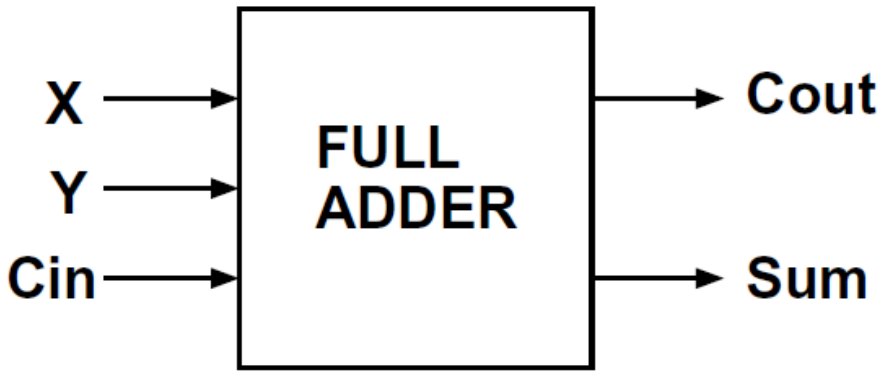
EXCLUSIVE OR: C = A ⊕ B

(A=B=1,C=0)

Truth table(真值表): An example

The behavior of a combinational logic circuit can be specified by a truth table

A truth table gives the circuit outputs for each combination of input values



(a) Full adder module

X	Y	Cin	Cout	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

(b) Truth Table

Truth table(真值表): An example

X	Y	Cin	Cout	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 = 1 when ...			
X	**Y**	**Cin**	
0	0	1	X'Y'Cin =1
0	1	0	X'Y Cin'=1
1	0	0	X Y'Cin'=1
1	1	1	X Y Cin =1

Sum = X'Y'Cin + X'YCin' + XY'Cin' + XYCin

Minterm expansion

Minterm(最小项)

Sum = m1 + m2 + m4 + m7 = Σm(1,2,4,7)

The truth table row for which Sum = 1

Truth table(真值表): An example

X	Y	Cin	Cout	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

Minterm expansion for Cout ?

Truth table(真值表): An example

X	Y	Cin	Cout	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

Cout = 1 when ...			
X	**Y**	**Cin**	
0	1	1	X' Y Cin =1
1	0	1	X Y' Cin=1
1	1	0	X Y Cin'=1
1	1	1	X Y Cin =1

Cout = X'YCin + XY'Cin + XYCin' + XYCin

Cout = m3 + m5 + m6 + m7 = Σ m(3,5,6,7)

Truth table(真值表): An example

X	Y	Cin	Cout	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 = 0 when ...

X	Y	Cin	
0	0	0	X + Y + Cin = 0
0	1	1	X + Y' + Cin' = 0
1	0	1	X' + Y + Cin = 0
1	1	0	X' + Y' + Cin = 0

Sum = (X+Y+Cin)(X+Y'+Cin')(X'+Y+Cin)(X'+Y'+Cin)

Maxterm(最大项)

Maxterm expansion

Sum = M0·M3·M5·M6 = $\Pi M(0, 3, 5, 6)$

The truth table rows for which Sum = 0

Truth table(真值表): An example

X	Y	Cin	Cout	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

Maxterm expansion for Cout?

Truth table(真值表): An example

X	Y	Cin	Cout	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

Cout = 0 when ...			
X	**Y**	**Cin**	
0	0	0	X + Y + Cin = 0
0	0	1	X + Y + Cin' = 0
0	1	0	X + Y' + Cin = 0
1	0	0	X' + Y + Cin = 0

Cout = (X+Y+Cin)(X+Y+Cin')(X+Y'+Cin)(X'+Y+Cin)

Cout = M0·M1·M2·M4 = $\prod M(0, 1, 2, 4)$

	Contents
1.1	Combinational Logic（组合逻辑）
1.2	**Boolean Algebra and Algebra Simplification**
1.3	Karnaugh Maps
1.4	Designing with NAND and NOR Gates
1.5	Hazards in Combinational Circuits
1.6	Flip-Flops and Latches
1.7	Mealy Sequential Circuit Design
1.8	Moore Sequential Circuit Design
1.9	Equivalent States and Reduction of State Tables
1.10	Sequential Circuit Timing
1.11	Tristate Logic and Busses

1.2 Boolean Algebra and Algebraic Simplification (布尔代数与代数式的化简)

- listed in dual pairs
- 0↔1, +↔*

Operations with 0 and 1	
X + 0 = X	X • 1 = X
X + 1 = 1	X • 0 = 0

Idempotent laws (同一律)	
X + X = X	X • X = X

Involution law (还原律)	
(X')' = X	

1.2 Boolean Algebra and Algebraic Simplification (布尔代数与代数式的化简)

Laws of complementarity(互补律)	
$X + X' = 1$	$X \cdot X' = 0$

Commutative laws (交换律)	
$X + Y = Y + X$	$XY = YX$

Associative laws (结合律)	
$(X + Y) + Z = X + (Y + Z) = X + Y + Z$	$(XY)Z = X(YZ) = XYZ$

Distributive laws (分配律)	
$X(Y+Z) = XY+XZ$	$X+YZ = (X+Y)(X+Z)$

1.2 Boolean Algebra and Algebraic Simplification (布尔代数与代数式的化简)

Simplication theorems(化简定理)

$XY + XY' = X$	$(X + Y)(X + Y') = X$
$X + XY = X$	$X(X + Y) = X$
$(X+Y')Y = XY$	$XY' + Y = X + Y$

DeMorgan's laws (狄摩根定律)

$(X + Y + Z + \ldots)' = X'Y'Z' \ldots$	$(XYZ \ldots)' = X' + Y' + Z' + \ldots$

$$[f(X1,X2,\ldots,Xn,0,1,+,\bullet)]' = f(X1',X2',\ldots,Xn',1,0,\bullet,+)$$

Duality(对偶式)

$(X + Y + Z + \ldots)^D = XYZ \ldots$	$(XYZ \ldots)^D = X + Y + Z + \ldots$

$$[f(X1,X2,\ldots,Xn,0,1,+,\bullet)]^D = f(X1,X2,\ldots,Xn,1,0,\bullet,+)$$

1.2 Boolean Algebra and Algebraic Simplification
(布尔代数与代数式的化简)

Theorem for multiplying out and factoring	
$(X + Y)(X' + Z) = XZ + X'Y$	$XY + X'Z = (X+Z)(X'+Y)$

Consensus theorem	
$XY + YZ + X'Z = XY + X'Z$	$(X+Y)(Y+Z)(X'+Z)$ $= (X +Y)(X' + Z)$

1.2 Boolean Algebra and Algebraic Simplification (布尔代数与代数式的化简)

DeMorgan's laws	
(X + Y + Z +...)' = X'Y'Z'...	(XYZ...)' = X' + Y' + Z' +...
[f(X1,X2,...,Xn,0,1,+,•)]' = f(X1',X2',...,Xn',1,0,•,+)	

To form the complement of a Boolean expression:

- replace each variable by its complement
- also replace 1 with 0, 0 with 1, OR with AND, and AND with OR
- add parentheses as required to assure the proper hierarchy of operations

1.2 Boolean Algebra and Algebraic Simplification (布尔代数与代数式的化简)

DeMorgan's laws	
$(X + Y + Z + \ldots)' = X'Y'Z'\ldots$	$(XYZ\ldots)' = X' + Y' + Z' + \ldots$
$[f(X1,X2,\ldots,Xn,0,1,+,\bullet)]' = f(X1',X2',\ldots,Xn',1,0,\bullet,+)$	

Example	F = X + E'K(C(AB+D')•1+WZ'(G'H+0)), F' = ?

1.2 Boolean Algebra and Algebraic Simplification (布尔代数与代数式的化简)

DeMorgan's laws	
(X + Y + Z +...)' = X'Y'Z'...	(XYZ...)' = X' + Y' + Z' +...
[f(X1,X2,...,Xn,0,1,+,•)]' = f(X1',X2',...,Xn',1,0,•,+)	

Example	F = X + E'K(C(AB+D')•1+WZ'(G'H+0)), F' = ?
X→X', 0→1, 1→0	X + E'•K•(C•(A•B+D')•1+W•Z'•(G'•H+0)) ↓ X' + E•K'•(C'•(A'•B'+D)•0+W'•Z•(G•H'+1))
+→•, •→+	↓ X' + E•K'•(C'•((A'+B')•D)•0+W'•Z•((G+H')•1)) ↓ X' + E•K'•((C'+(A'+B')•D+0)•(W'+Z+(G+H')•1)) ↓ X'•(E+K'+(C'+(A'+B')•D+0)•(W'+Z+(G+H')•1))

before change
after change

Example

- (A+BC)’=A’(B’+C’)
- (A+BC)’≠A’B’+C’

- A=1,B=1,C=0
- (A+BC)’=0
- A’(B’+C’)=0
- A’B’+C’=1

1.2 Boolean Algebra and Algebraic Simplification

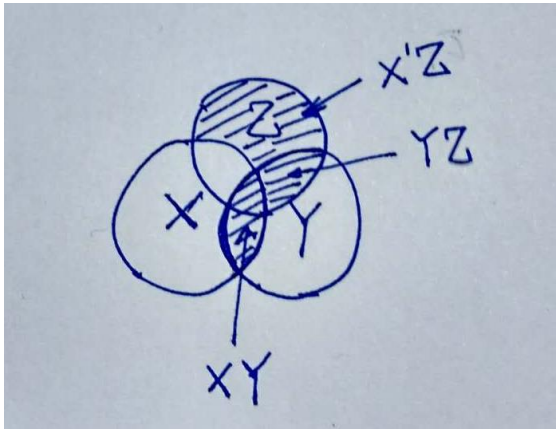
1. Combining terms: XY + XY' = X

ABC'D' + ABCD' = ABD' [X = ABD', Y = C]

Cout = X'YCin + XY'Cin + XYCin' + XYCin
= (X'YCin + XYCin) + (XY'Cin + XYCin) + (XYCin' + XYCin)
= YCin + XCin + XY [X = X + X]

(A+BC)(D+E') + A'(B'+C')(D+E')
= (A+BC)(D+E') + (A+BC)'(D+E')
= D+ E'

1.2 Boolean Algebra and Algebraic Simplification



Venn diagram

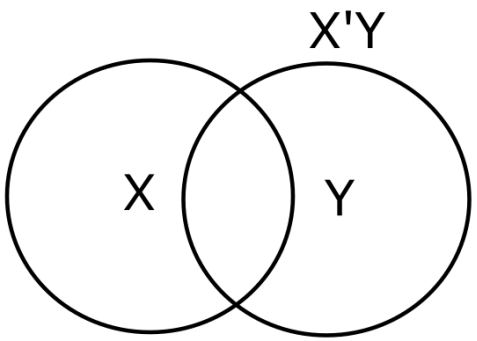
Consensus theorem: XY + X'Z + YZ = XY + X'Z

Proof: XY + X'Z + YZ = XY + X'Z + (X + X')YZ
= XY + X'Z + XYZ + X'YZ
= (XY + XYZ) + (X'Z + X'YZ)
= XY(1 + Z) + X'Z(1 + Z)
= XY + X'Z

2. Eliminating terms: X + XY = X
XY + X'Z + YZ = XY + X'Z

A'B+A'BC = A'B	[X = A'B]
A'BC' + BCD + A'BD = A'BC' + BCD	[X = C, Y = BD, Z = A'B]

1.2 Boolean Algebra and Algebraic Simplification



3. Eliminating literals: X + X'Y = X + Y

A'B + A'B'C'D' + ABCD' = A'(B + B'C'D') + ABCD'
= A'(B + C'D') + ABCD'
= B(A' + ACD') + A'C'D'
= B(A' + CD') + A'C'D'
= A'B + BCD' + A'C'D'

X	Y	X+X'Y	X+Y
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

1.2 Boolean Algebra and Algebraic Simplification

4. Adding redundant terms:

WX + XY + X'Z' + WY'Z' (Add WZ' by the consensus theorem.)
= WX + XY + X'Z' + WY'Z' + WZ' (WX+X'Z'=WX+X'Z'+WZ')
= WX + XY + X'Z' + WZ'(1+Y') (Eliminate WY'Z'.)
= WX + XY + X'Z' + WZ' (Eliminate WZ'.)
= WX +XY + X'Z'

(A + B + D)(A + B' + C')(A' + B + D')(A' + B + C')
= (A + (B+D)(B'+C'))(A' + B + C'D') (X+YZ)=(X+Y)(X+Z)
= (A + BC' + B'D)(A' + B + C'D') (X+Y)(X'+Z)=XZ+X'Y
= A(B + C'D') + A'(BC' + B'D) (X+Y)(X'+Z)=XZ+X'Y
= AB + AC'D' + A'BC' + A'B'D

1.2 Boolean Algebra and Algebraic Simplification

Theorems with exclusive-OR	
X XOR 0 = X	**A XOR B = AB' + A'B**
X XOR 1 = X'	
X XOR X = 0	
X XOR X' = 1	
X XOR Y = Y XOR X	(commutative law)
(X XOR Y) XOR Z = X XOR (Y XOR Z) = X XOR Y XOR Z	(associative law)
X(Y XOR Z) = XY XOR XZ	(distributive law)
(X XOR Y)' = X XOR Y' = X' XOR Y = XY + X'Y'	

Sum = X'Y'Cin + X'YCin' + XY'Cin' + XYCin	**(Full adder)**
= X'(Y'Cin + YCin') + X(Y'Cin' + YCin) = X'(Y XOR Cin) + X(Y XOR Cin)' = X XOR Y XOR Cin	

	Contents
1.1	Combinational Logic（组合逻辑）
1.2	Boolean Algebra and Algebra Simplification
1.3	**Karnaugh Maps**
1.4	Designing with NAND and NOR Gates
1.5	Hazards in Combinational Circuits
1.6	Flip-Flops and Latches
1.7	Mealy Sequential Circuit Design
1.8	Moore Sequential Circuit Design
1.9	Equivalent States and Reduction of State Tables
1.10	Sequential Circuit Timing
1.11	Tristate Logic and Busses

1.3 Karnaugh maps (卡诺图)

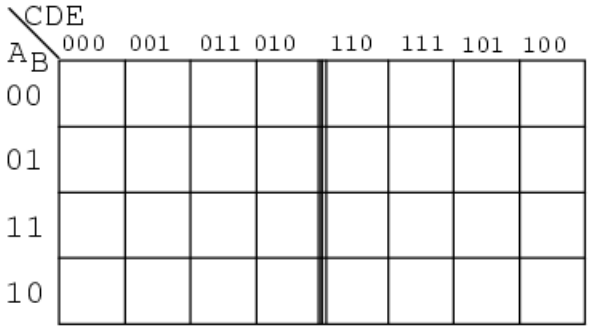
Four-variable Karnauph map

\AB CD\	00	01	11	10
00	m0	m4	m12	m8
01	m1	m5	m13	m9
11	m3	m7	m15	m11
10	m2	m6	m14	m10

minterm

\AB CD\	00	01	11	10
00	1	0	0	0
01	0	0	0	0
11	0	0	1	0
10	0	0	0	0

F= m0 + m15 = A'B'C'D' + ABCD



Five-variable Karnauph map

1.3 Karnaugh maps (卡诺图)

Four-variable Karnauph map

\AB CD\	00	01	10	11

\AB CD\	00	01	11	10
00	m0	m4	m12	m8
01	m1	m5	m13	m9
11	m3	m7	m15	m11
10	m2	m6	m14	m10

\AB CD\	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	1	1	0
10	0	0	0	0

The values along the edge of K-map are ordered so that adjacent squares on the map differ in only one variable

F= A'BCD + ABCD = BCD

Two 1's in adjacent squares can be combined by eliminating one variable using **XY + XY' = X**

1.3 Karnaugh maps (卡诺图)

\AB CD\	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

A’BC’D + ABC’D = BC’D

A'BCD + ABCD = BCD

BC’D +BCD = BD

1.3 Karnaugh maps (卡诺图)

\AB CD\	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

The top and bottom rows of the map are adjacent

The first and last columns of the map are adjacent

F= A'B'CD' + AB'CD' = B'CD'

1.3 Karnaugh maps (卡诺图)

Four-variable Karnauph map

\AB CD\	00	01	11	10
00	0	0	0	0
01	0	0	X	X
11	0	1	1	0
10	0	0	0	0

X: We don't care whether the minterm is present or not

Don't cares arise under two conditions:

1. The input condition corresponding to the don't care can **never occur**
2. The input combination can occur, but the circuit output is **not specified** for this input condition

1.3 Karnaugh maps (卡诺图)

\AB CD\	00	01	11	10
00	1	0	0	1
01	0	1	0	0
11	1	1	X	1
10	1	1	X	1

Four corner terms combine to give B'D'

A'BD

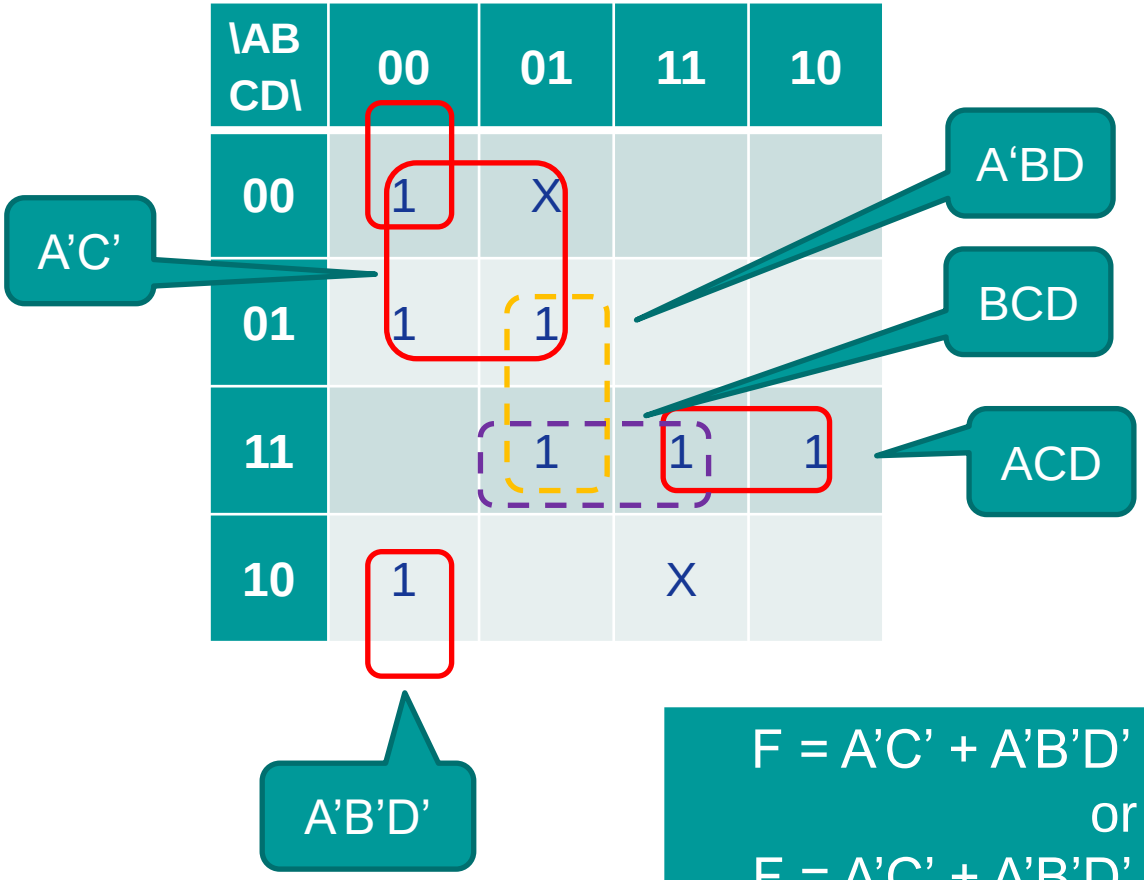
C

A'BC'D + A'BCD = A'B'D

(A'B'C'D' + AB'C'D') + (A'B'CD' + AB'CD') = B'C'D' +B'CD' = B'D'

Selection of prime implicants

A group of one, two, four, or eight adjacent 1's on a map represents a **prime implicant** （质蕴涵项）if it cannot be combined with another group of 1's to eliminate a variable

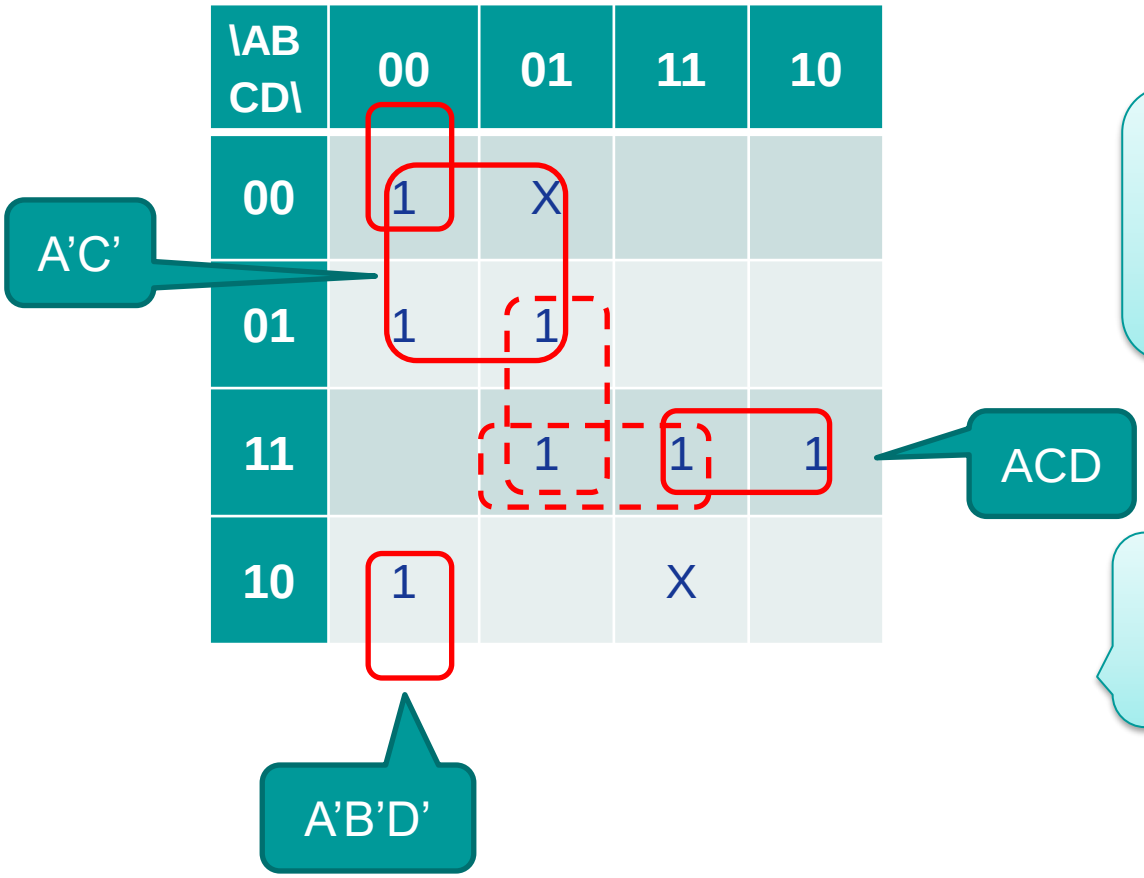


Prime implicants:
A'C',
ACD,
A'B'D',
A'BD,
BCD

F = A'C' + A'B'D' + ACD + A'BD
or
F = A'C' + A'B'D' + ACD + BCD

Selection of prime implicants

Essential prime implicant (基本质蕴涵项)

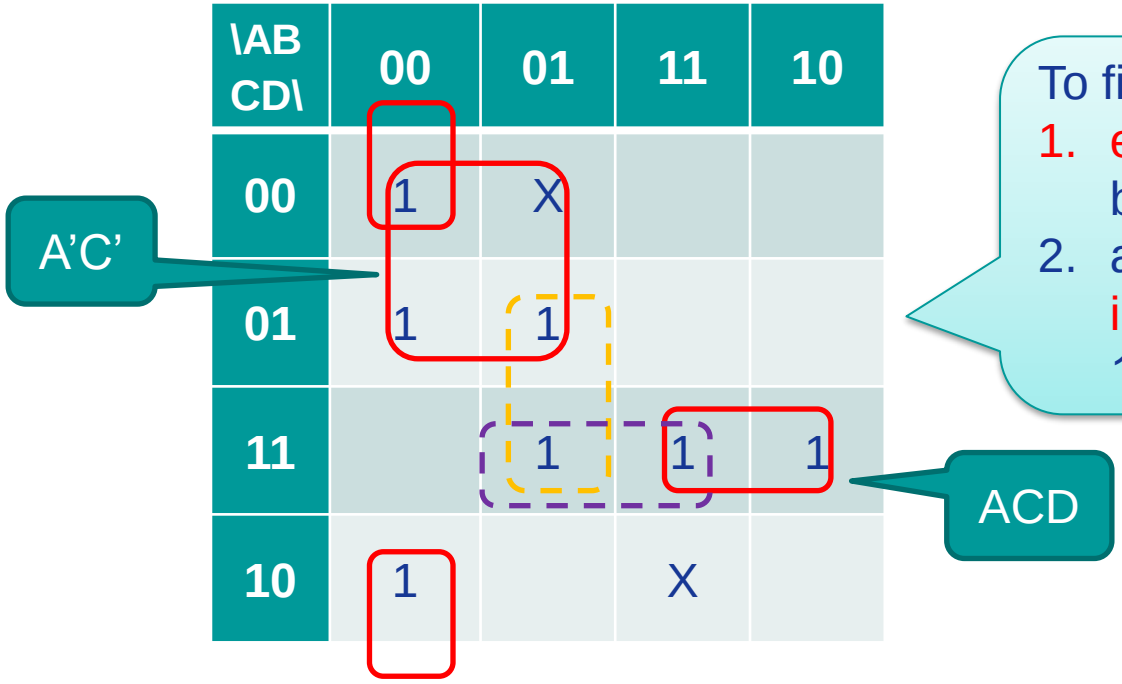


A prime implicant is **essential** if it contains a 1 that is not contained in any other prime implicant

Essential prime implicants:
A'C', ACD, A'B'D'

Selection of prime implicants

The minimum sum-of-products(SOP, 与或表达式) representation of a function consists of a sum of prime implicants



To find a minimum SOP from a map:

1. essential prime implicants should be looped first, and then
2. a minimum number of prime implicants to cover the remaining 1’s should be looped

F = A’C’ + A’B’D’ + ACD + A’BD
or
F = A’C’ + A’B’D’ + ACD + BCD

A'C', ACD, A'B'D' are essential

1.3 Karnaugh maps (卡诺图)

\AB CD\	00	01	11	10
00	1	0	0	1
01	0	1	0	0
11	1	1	X	1
10	1	1	X	1

Minimum SOP (最小与或表达式)

F = C + B'D' + A'BD

Minimum POS（最小或与表达式）？

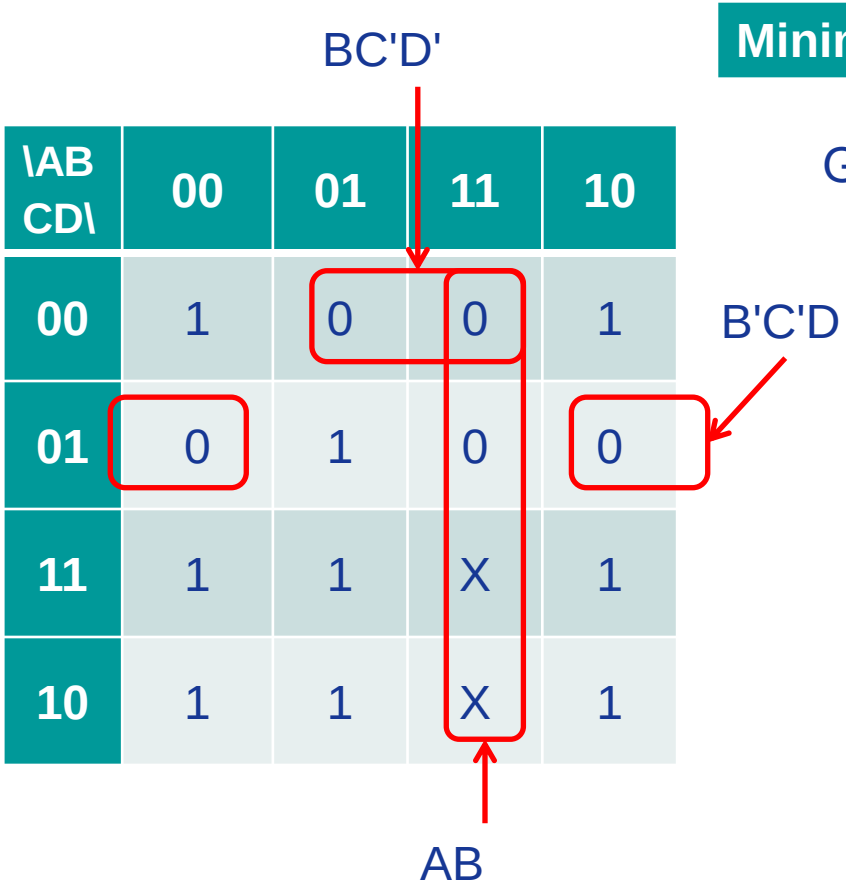
G = (X1+X2+...)•(X3+X4+...)•(...+...)...

DeMorgan's laws

G' = X1'•X2'•...+X3'•X4'•...+... •...+...

SOP

1.3 Karnaugh maps (卡诺图)



\AB CD\	00	01	11	10
00	1	0	0	1
01	0	1	0	0
11	1	1	X	1
10	1	1	X	1

Minimum POS（最小或与表达式）

$G = (X1+X2+...)\bullet(X3+X4+...)\bullet(...+...)...$

DeMorgan's laws

$G' = X1'\bullet X2'\bullet...+X3'\bullet X4'\bullet...+... \bullet...+...$

Loop the 0's

$G' = BC'D' + B'C'D + AB$

DeMorgan's laws

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

1.3.1 Simplification using map-entered variables

Partial Truth Table for a Six-Variable Function

Inputs						Output
A	B	C	D	E	F	G
0	0	0	0	X	X	1
0	0	0	1	X	X	X
0	0	1	0	X	X	1
0	0	1	1	X	X	1
0	1	0	1	1	X	1
0	1	1	1	1	X	1
1	0	0	1	X	1	1
1	0	1	0	X	X	X
1	0	1	1	X	X	1
1	1	0	1	X	X	X
1	1	1	1	X	X	1

The input combinations not specified result in 0

1.3.1 Simplification using map-entered variables

Inputs						Output
A	B	C	D	E	F	G
0	0	0	0	X	X	1
0	0	0	1	X	X	X
0	0	1	0	X	X	1
0	0	1	1	X	X	1
0	1	0	1	1	X	1
0	1	1	1	1	X	1
1	0	0	1	X	1	1
1	0	1	0	X	X	X
1	0	1	1	X	X	1
1	1	0	1	X	X	X
1	1	1	1	X	X	1

\AB CD\	00	01	11	10
00	1			
01	X	E	X	F
11	1	E	1	1
10	1			X

G

E and F are the input variables with the most number of don't cares

- a Karnaugh map can be formed with A, B,C,D
- E and F can be entered in the squares in the map

1.3.1 Simplification using map-entered variables

\AB CD\	00	01	11	10
00	1			
01	X	E	X	F
11	1	E	1	1
10	1			X

G

\AB CD\	00	01	11	10
00	1·m0			
01	X	E·m5	X	F·m9
11	1·m3	E·m7	1·m15	1·m11
10	1·m2			X

G(A,B,C,D,E,F) = m0 + m2 + m3 + E·m5 + E·m7 + F·m9 + m11 + m15
(+ don't care terms)

1.3.1 Simplification using map-entered variables

\AB CD\	00	01	11	10
00	1			
01	X	E	X	F
11	1	E	1	1
10	1			X

=

- The minimum sum obtained whatever the map-entered variables, i.e., E & F are
- EF can be 00, 01, 10, 11

G

+

- The minimum sum obtained when E = 1 and whatever the other map-entered variables (F) is
- F can be 0, 1
- The 1's on the map are treated as X (They have been considered in the first term)

+

- The minimum sum obtained when F = 1 and whatever the other map-entered variables (E) is
- E can be 0, 1
- The 1's on the map are treated as X again

All the cases have been considered

1.3.1 Simplification using map-entered variables

\AB CD\	00	01	11	10
00	1			
01	X	E	X	F
11	1	E	1	1
10	1			X

G

=

\AB CD\	00	01	11	10
00	1			
01	X		X	
11	1		1	1
10	1			X

Treat E=0,F=0

+

\AB CD\	00	01	11	10
00	X			
01	X	1	X	
11	X	1	X	X
10	X			X

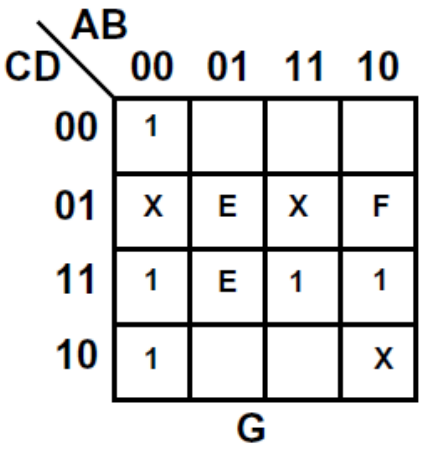
Treat E=1,F=0

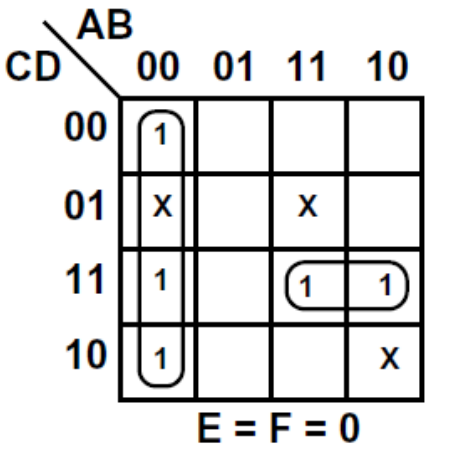
+

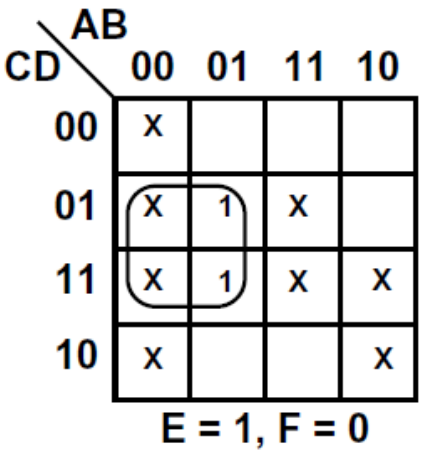
\AB CD\	00	01	11	10
00	X			
01	X		X	1
11	X		X	X
10	X			X

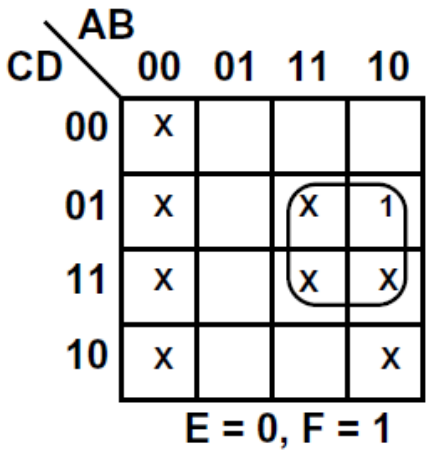
Treat E=0,F=1

1.3.1 Simplification using map-entered variables









$$G = MS_0 + EMS_1 + FMS_2$$

$$= A'B' + ACD + EA'D + FAD$$

It is not required to draw a map for E=1, F=1 (WHY?)

Because all the cases have been considered

\AB CD\	00	01	11	10
00	1			
01	X	E	X	F
11	1	E	1	1
10	1			X

G

=

\AB CD\	00	01	11	10
00	1			
01	X		X	
11	1		1	1
10	1			X

E=0,F=0

+

\AB CD\	00	01	11	10
00	X			
01	X	1	X	
11	X	1	X	X
10	X			X

E=1,F=0

+

\AB CD\	00	01	11	10
00	X			
01	X		X	1
11	X		X	X
10	X			X

E=0,F=1

+

\AB CD\	00	01	11	10
00	X			
01	X	1	X	1
11	X	1	X	X
10	X			X

E=1,F=1
$MS_3 = D$

G = A'B' + ACD + EA'D + FAD + EF·D

When E = 1 and F = 1, EF·D has already been covered by EA'D + FAD (A'D + AD = D)

1.3.1 Simplification using map-entered variables

\AB CD\	00	01	11	10
00	1			
01	X	E	X	F
11	1	E	1	1
10	1			X

G

\AB CD\	00	01	11	10
00	1			
01	X		X	
11	1		1	1
10	1			X

E=0,F=0

\AB CD\	00	01	11	10
00	X			
01	X	1	X	
11	X	1	X	X
10	X			X

E=1,F=0

\AB CD\	00	01	11	10
00	X			
01	X		X	1
11	X		X	X
10	X			X

E=0,F=1

\AB CD\	00	01	11	10
00	X			
01	X	1	X	1
11	X	1	X	X
10	X			X

E=1,F=1

Not so sure?
Let us have a try

\AB CD\	00	01	11	10
00	1			
01	X	E	X	F
11	1	E	1	1
10	1			X

G

\AB CD\	00	01	11	10
00	1			
01	X		X	
11	1		1	1
10	1			X

E=0,F=0
MS0 = A'B'+ACD

\AB CD\	00	01	11	10
00	X			
01	X	1	X	
11	X	1	X	X
10	X			X

E=1,F=0
MS1 = A'D

\AB CD\	00	01	11	10
00	X			
01	X		X	1
11	X		X	X
10	X			X

E=0,F=1
MS2 = AD

\AB CD\	00	01	11	10
00	X			
01	X	1	X	1
11	X	1	X	X
10	X			X

E=1,F=1
MS3 = D

G = MS0 + EF' •MS1+E'F•MS2 + EF•MS3
= A'B'+ACD + EF' •A'D+E'F•AD + EF• D

Can it be further simplified to
G = A'B' + ACD + EA'D + FAD?

G = A’B’ + ACD + EF’ •A’D + E’F•AD + EF• D

= A’B’ + ACD + A’DE + ADF

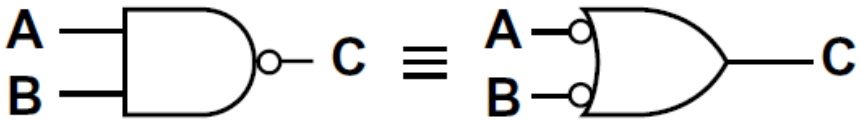
EF’ •A’D+E’F•AD + EF• D

\AD EF\	00	01	11	10
00				
01			1	
11		1	1	
10		1		

	Contents
1.1	Combinational Logic（组合逻辑）
1.2	Boolean Algebra and Algebra Simplification
1.3	Karnaugh Maps
1.4	**Designing with NAND and NOR Gates**
1.5	Hazards in Combinational Circuits
1.6	Flip-Flops and Latches
1.7	Mealy Sequential Circuit Design
1.8	Moore Sequential Circuit Design
1.9	Equivalent States and Reduction of State Tables
1.10	Sequential Circuit Timing
1.11	Tristate Logic and Busses

1.4 Designing With NAND and NOR gates

NAND:



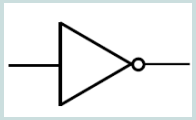
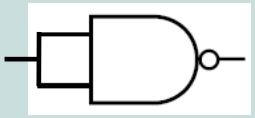
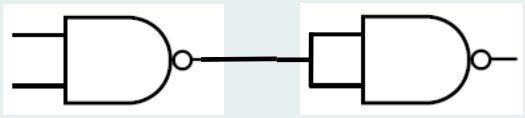
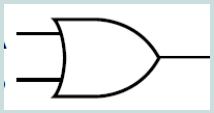
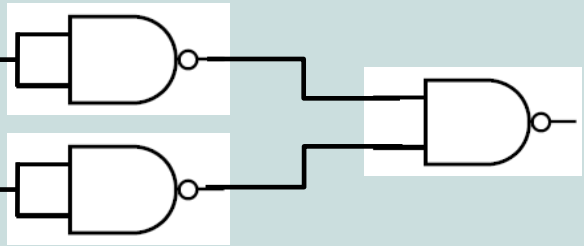
$C = (AB)' = A' + B'$

NOR:

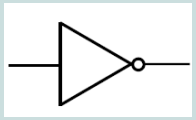
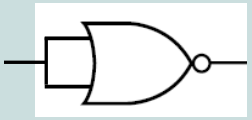
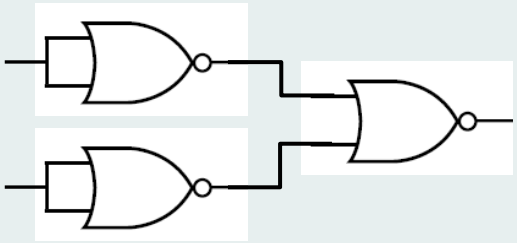
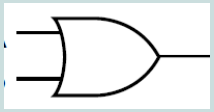
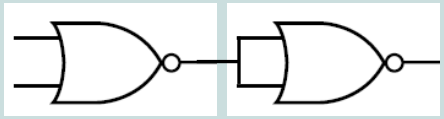
$C = (A+B)' = A'B'$

Any logic function can be realized using
- only NAND gates or
- only NOR gates

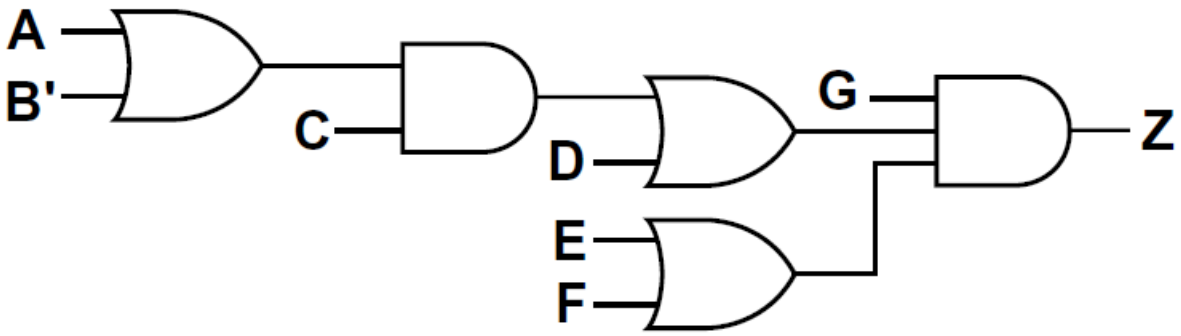
Functinally Complete Gates

NOT, AND, and OR gates	NAND gate equivalent circuits
A  A'	A  A'
A B  A·B	A B  ((A·B)')'=A·B
A B  A+B	A  (A'·B')'=A+B B

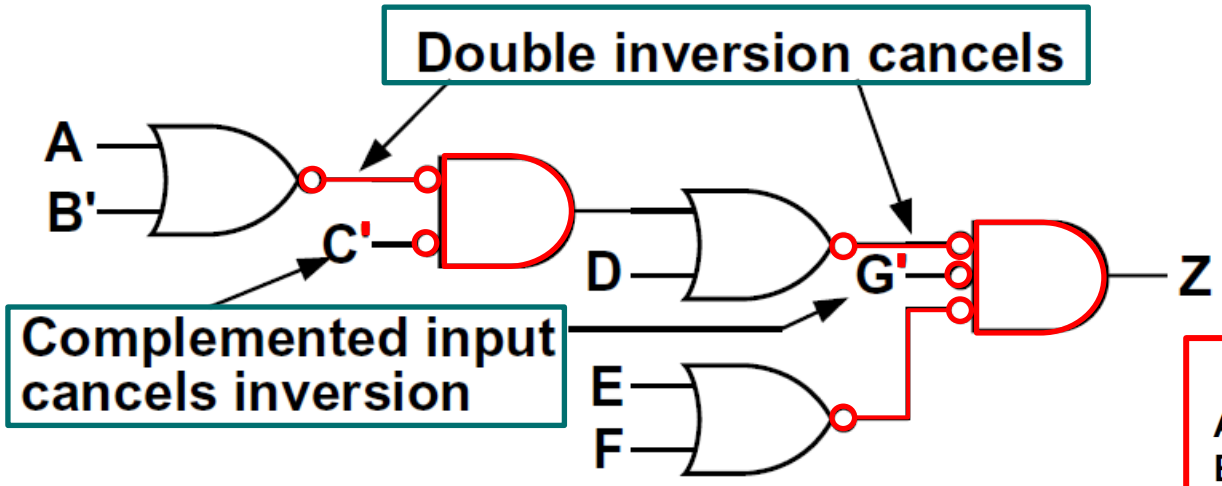
Functinally Complete Gates

NOT, AND, and OR gates	NOR gate equivalent circuits
A  A'	A  A'
A B  A·B	A B  (A'+B')'=AB
A B  A+B	A B  ((A+B)')'=A+B

Conversion to NOR gates

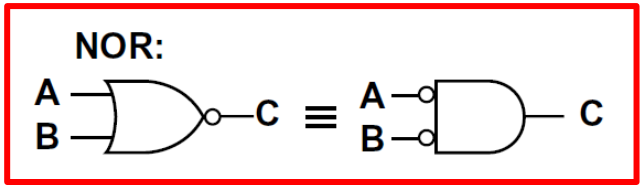


(a) AND-OR network

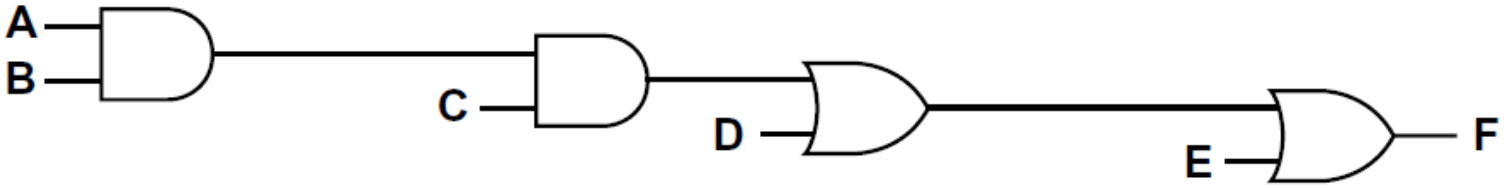


(b) Equivalent NOR-gate network

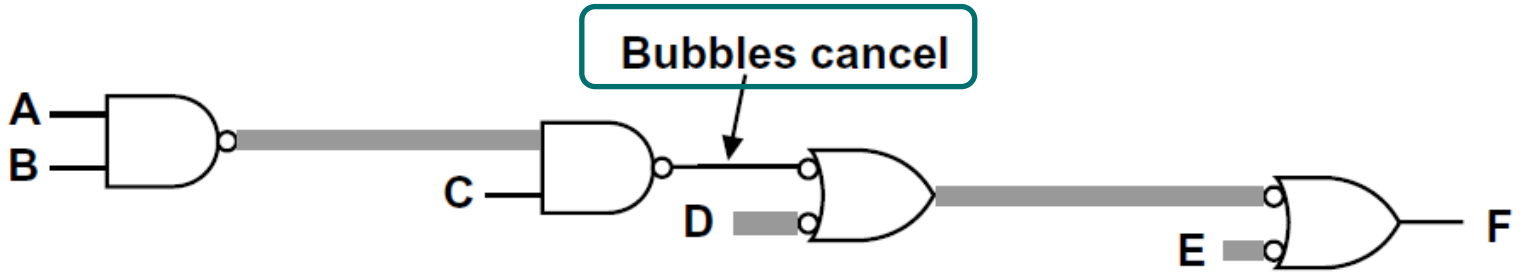
(A+B)'=A'B'



Conversion to NAND gates

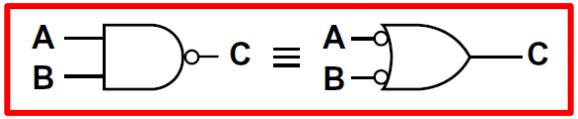


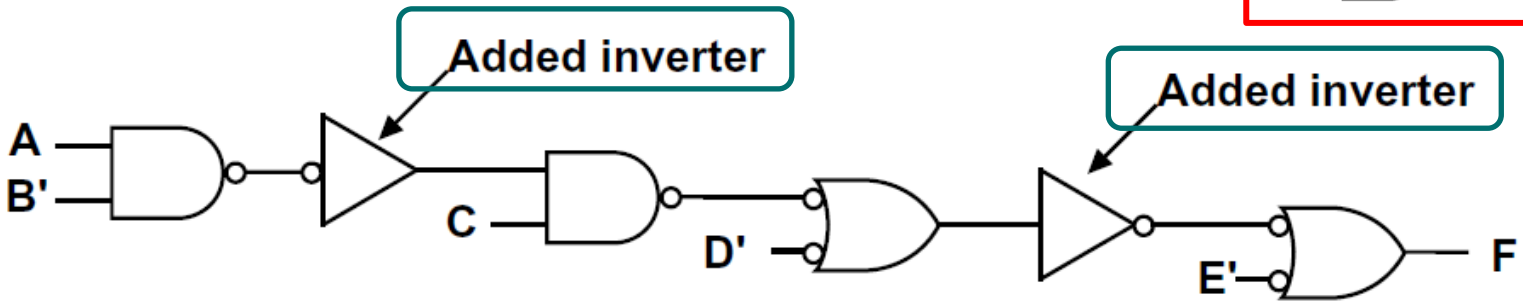
(a) AND_OR network



(b) First step in NAND conversion

(AB)'=A'+B'

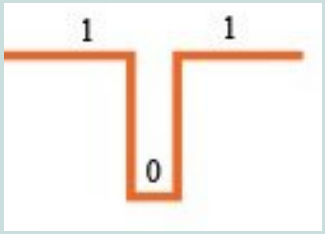
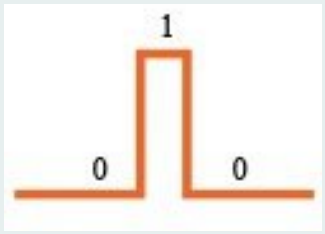




(c) Completed conversion

	Contents
1.1	Combinational Logic（组合逻辑）
1.2	Boolean Algebra and Algebra Simplification
1.3	Karnaugh Maps
1.4	Designing with NAND and NOR Gates
1.5	**Hazards in Combinational Circuits**
1.6	Flip-Flops and Latches
1.7	Mealy Sequential Circuit Design
1.8	Moore Sequential Circuit Design
1.9	Equivalent States and Reduction of State Tables
1.10	Sequential Circuit Timing
1.11	Tristate Logic and Busses

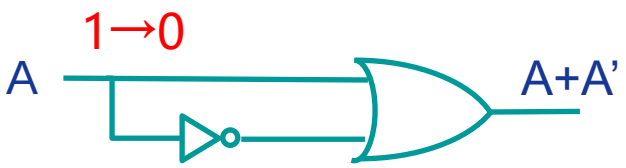
1.5 Hazards in Combinational Circuits

Hazards		
Static 1-hazard (静态1冒险)	A circuit output may momentarily go to 0 when it should remain a constant 1	
Static 0-hazard (静态0冒险)	A circuit output may momentarily go to 1 when it should remain a constant 0	
Dynamic hazard (动态冒险)	When the output is supposed to change from 0 to 1 (or 1 to 0), the output may change three or more times	

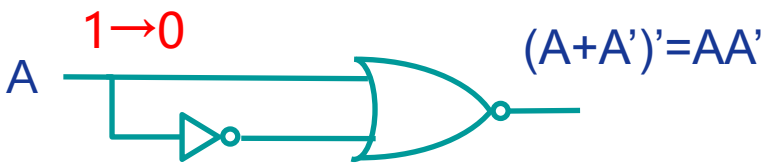
Steady-state output of the circuit is correct, however, a switching variation appears at the circuit output when the input is changed

1.5 Hazards in Combinational Circuits

Hazards	
Static 1-hazard	A circuit output may momentarily go to 0 when it should remain a constant 1
Static 0-hazard	A circuit output may momentarily go to 1 when it should remain a constant 0
Dynamic hazard	When the output is supposed to change from 0 to 1 (or 1 to 0), the output may change three or more times



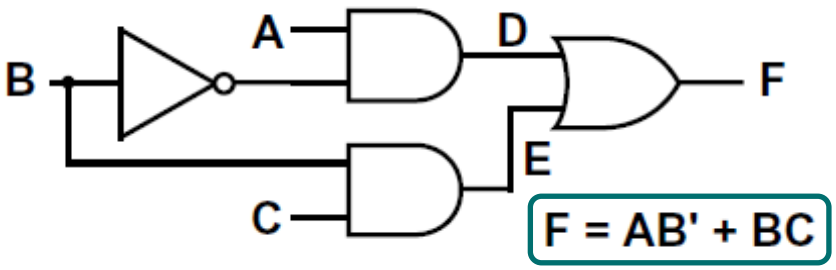
Simple circuit with static 1-hazard



Simple circuit with static 0-hazard

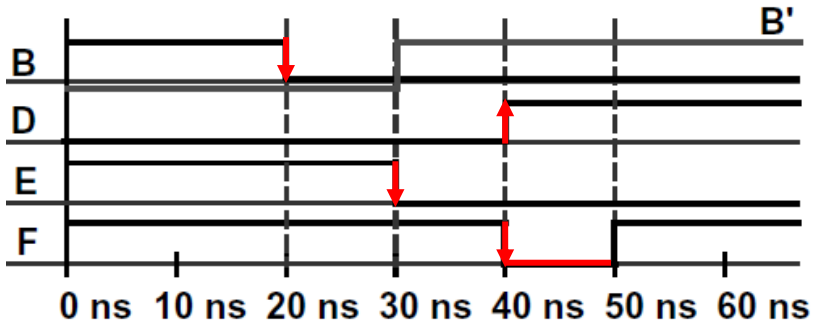
1.5 Hazards in Combinational Circuits

Example: Circuit with 1-hazard



Assume each gate has a propagation delay of 10ns

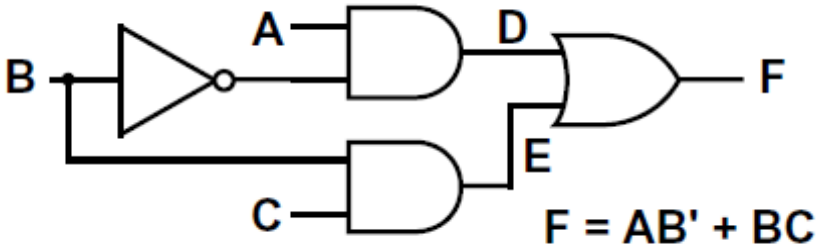
Assume A = C = 1,
F = B' + B = 1

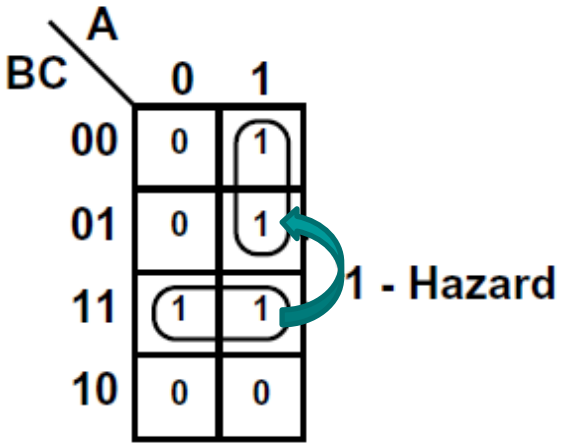


When B changes from 1 to 0, E changes to 0 before D changes to 1, causing F to monentarily change to 0

1.5 Hazards in Combinational Circuits

Example: Circuit with 1-hazard

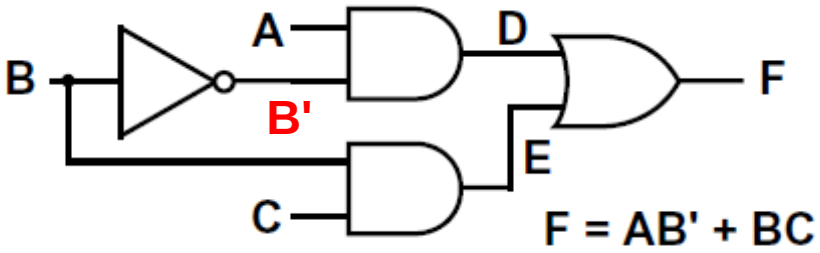




A static 1-hazard occurs in a SOP implementation when two minterms differing by only one input variable are not covered by the same product term

1.5 Hazards in Combinational Circuits

Circuit with 1-hazard



\AB' BC\	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	1	1	1	1
10	0	0	1	0

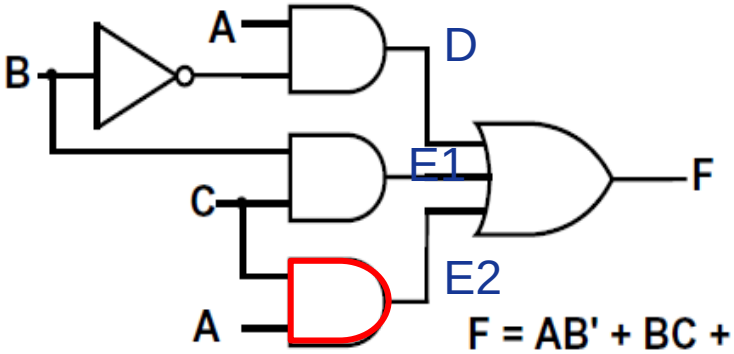
When AB'BC = 1001, F = 0

AB'BC

1011 → 1101

1011 → 1001 → 1101

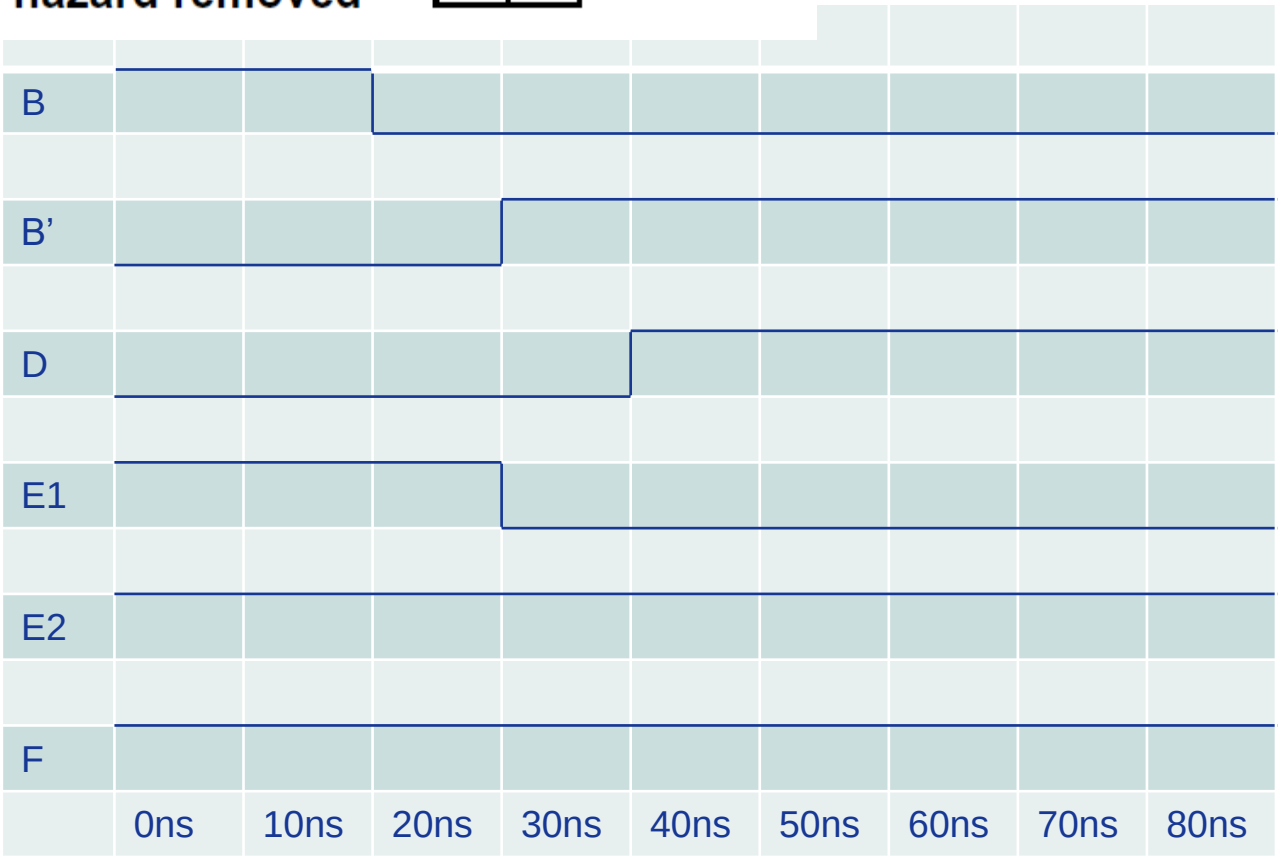
Circuit with hazard removed



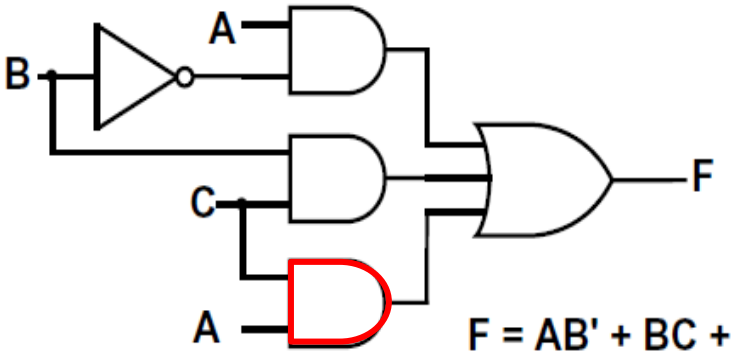
F = AB' + BC + AC

(c) Network with hazard removed

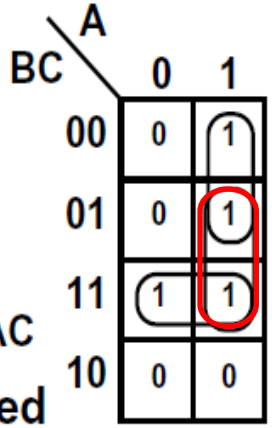
BC \ A	0	1
00	0	1
01	0	1
11	1	1
10	0	0



Circuit with hazard removed



(c) Network with hazard removed



\AB' BC\	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	1	1	1	1
10	0	0	1	0

When AB'BC = 1001, F = 1