Review of logic design fundamentals

K-map – map entered variables

- Functions – more than 5 variables – simplification using map entered variables.

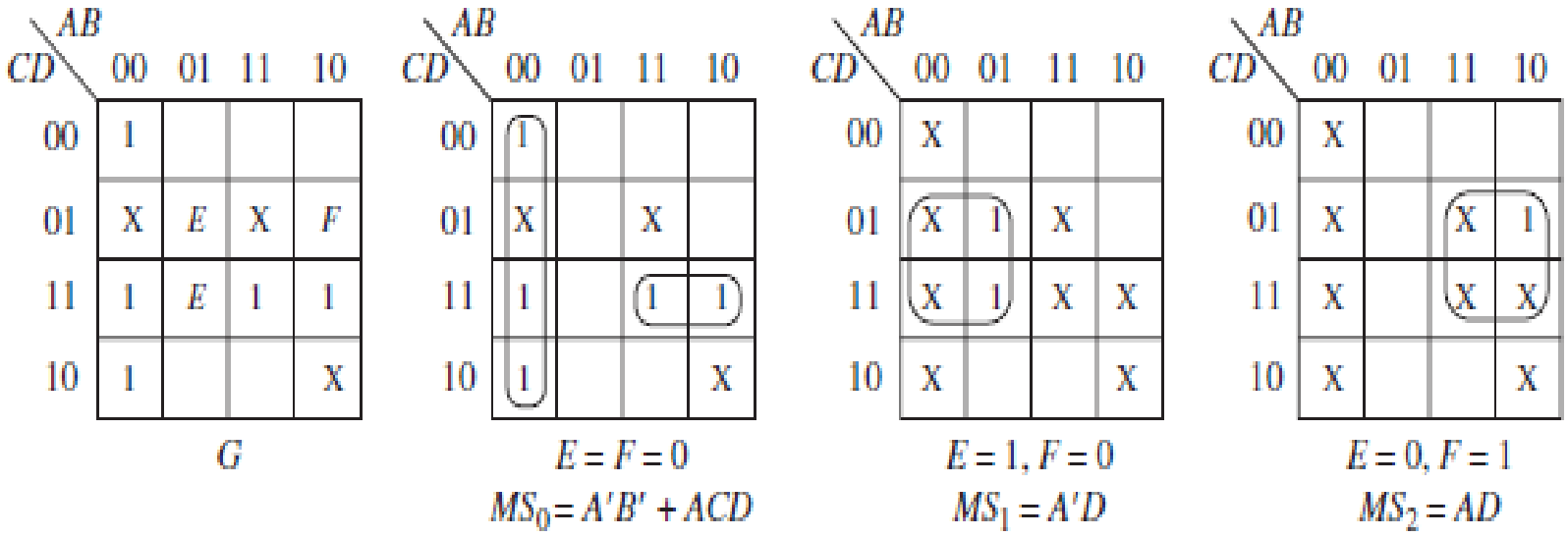
A	B	C	D	E	F	G
0	0	0	0	X	X	1
0	0	0	1	X	X	X
0	0	1	0	X	X	1
0	0	1	1	X	X	1
0	1	0	1	1	X	1
0	1	1	1	1	X	1
1	0	0	1	X	1	1
1	0	1	0	X	X	X
1	0	1	1	X	X	1
1	1	0	1	X	X	X
1	1	1	1	X	X	1

- Partial truth table for a 6 variable function
- Unspecified combinations – output = 0.

K-map – map entered variables

- E and F are input variables with greatest number of don’t cares.
- K map - formed with A, B,C ,D - remaining two variables entered inside the map.
- E appears in a square E = 1 – corresponding minterm is present in the function G.
- F appears in a square F = 1.

K-map – map entered variables



Simplification using map entered variables

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

K-map – map entered variables

$$F = MS_0 + P_1 MS_1 + P_2 MS_2 + \cdots$$

where

- MS_0 is the minimum sum obtained by setting $P_1 = P_2 = \cdots = 0$.
- MS_1 is the minimum sum obtained by setting $P_1 = 1, P_j = 0\ (j \neq 1)$, and replacing all 1s on the map with don't cares.
- MS_2 is the minimum sum obtained by setting $P_2 = 1, P_j = 0\ (j \neq 2)$, and replacing all 1s on the map with don't cares.

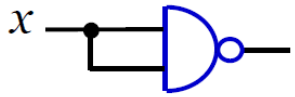
$$G = A'B' + ACD + EA'D + FAD$$

Design with NAND and NOR gates

- Concept of universality: Gates which can implement any Boolean function without the need to use any other types of gates.
- NAND and NOR universal gates.
- Use NAND and NOR alone to implement the basic gates.

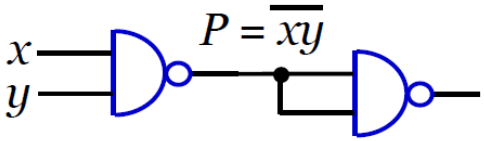
Design with NAND and NOR gates

NOT or **INV**



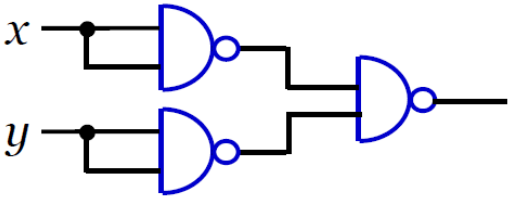
$F = \overline{x.x} = \overline{x}$

AND



$F = \overline{\overline{xy}} = xy$

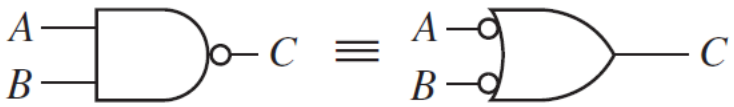
OR



$F = \overline{\overline{x} \cdot \overline{y}} = \overline{\overline{x}} + \overline{\overline{y}} = x + y$

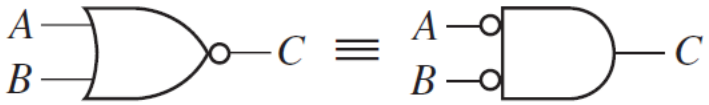
Design with NAND and NOR gates

NAND:



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

NOR:



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

Implementation of NAND and NOR easier than that of AND and OR gates.

Bubble – complement.

Design with NAND and NOR gates

AND – NAND - adding an inversion bubble at the output.

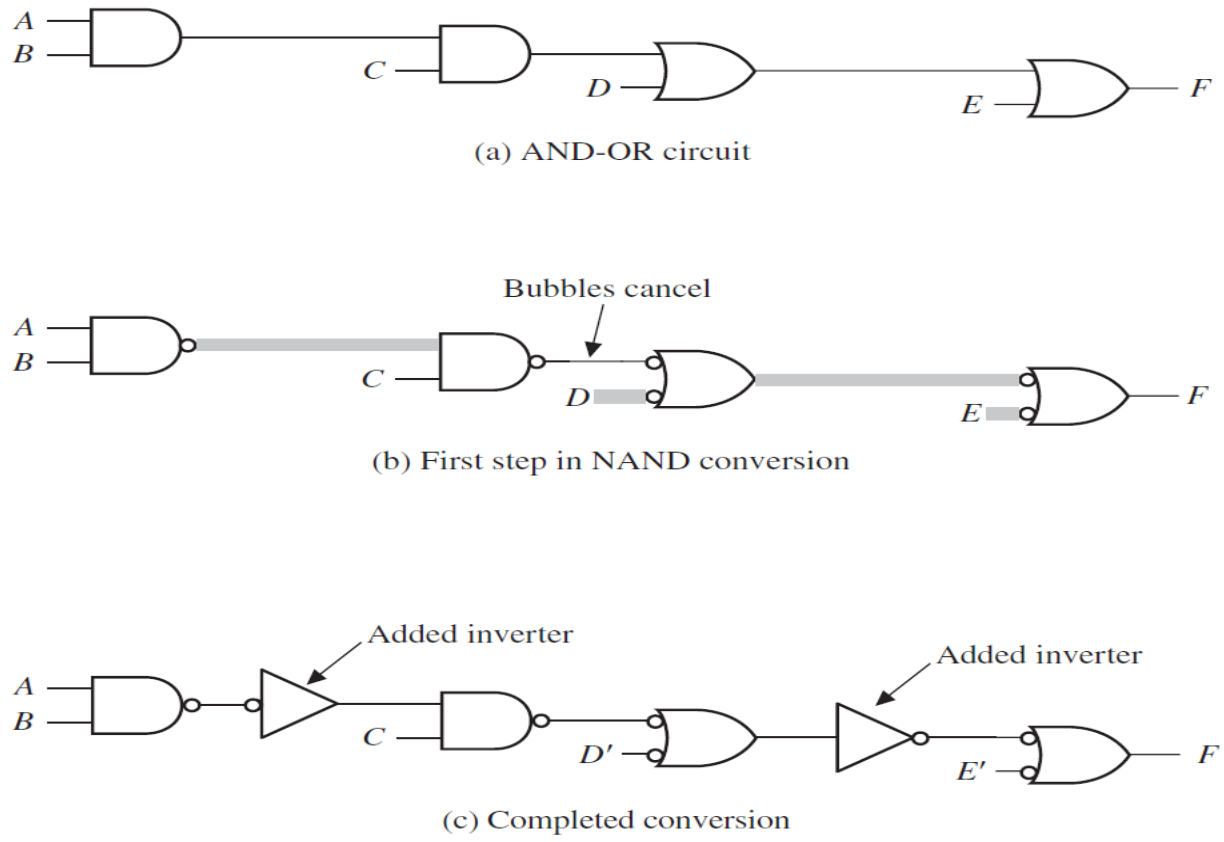
OR – NAND – adding inversion bubble at the inputs.

Inverter output drives inverted input – no further action required.

Non inverted gate output drives an inverted gate input or vice versa – insert an inverter.

Variable drives inverted input - complement the variable.

Design with NAND and NOR gates



Conversion of AND-OR circuit to NAND gate

Flip-flops and Latches

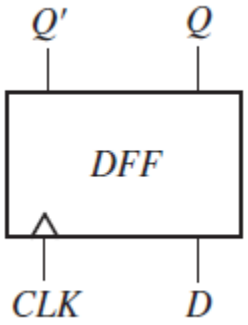
Sequential circuits – flip-flops as storage devices.

Types of flip-flops:

D flip-flops
J-K flip-flops
T flip-flops

Flip-flops and Latches

Clocked D flip-flop with rising edge trigger



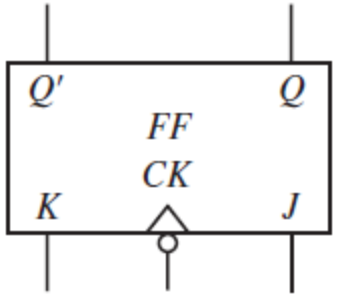
D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

$$Q^+ = D$$

- State change in response to the rising edge of the clock input.
- Q+ - next state of the Q output after the active edge of the clock.
- D – input before the active edge.

Flip-flops and Latches

J-K flip-flop and its truth table



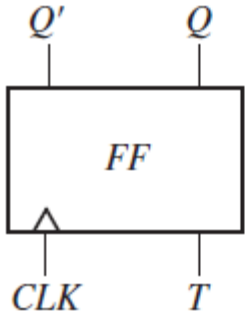
J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$$Q^+ = JQ' + K'Q.$$

- State change occurs following the falling edge of the clock.

Flip-flops and Latches

Clocked T flip-flop

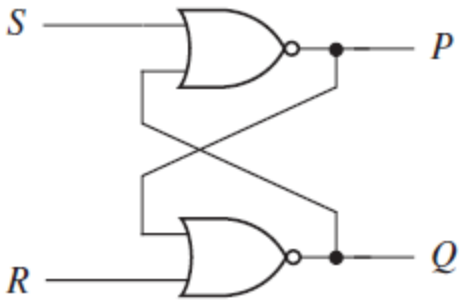


T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

$$Q^+ = QT' + Q'T = Q \oplus T$$

- State change occurs following the falling edge of the clock.

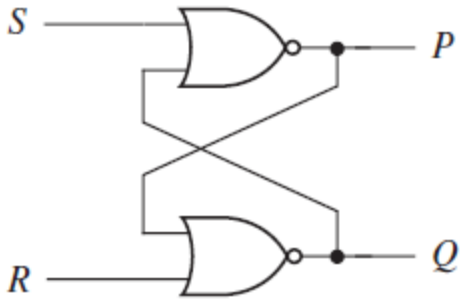
Flip-flops and Latches



$$Q^+ = S + R'Q.$$

Flip-flops and Latches

S-R latch

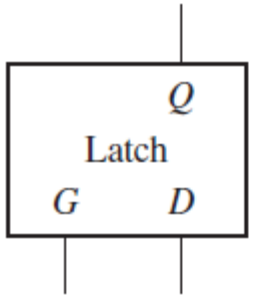


S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

- Un-clocked flip-flop – SR latch
- Characteristic equation?

Flip-flops and Latches

Transparent D latch



G	D	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$Q^+ = GD + G'Q.$$

- No response to input changes unless G = 1.
- Level sensitive D flip-flop