ITM1010
Computer and Communication Technologies

Lecture #5

Part I: Introduction to Computer Technologies

K-Map, Combination and Sequential Logic Circuits

Product-Of-Sum Configuration

Direct extraction of POS from truth table:

- For each 0 of "F", write the corresponding term in the form of sum of complements of inputs.
- "F" equals the product of the sums.

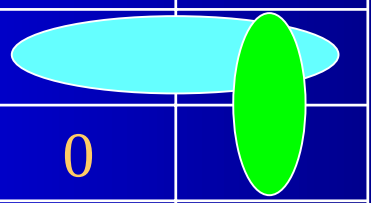
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

$$F = (A + B + C)(A + B + \overline{C})(A + \overline{B} + \overline{C})(\overline{A} + B + C)(\overline{A} + \overline{B} + C)(\overline{A} + \overline{B} + \overline{C})$$

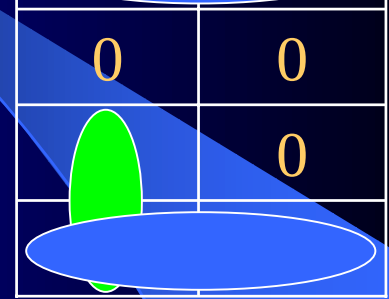
3-input K-map

	$\overline{C}$	C
$\overline{A}\,\overline{B}$	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}\,\overline{B}C$
$\overline{A}B$	$\overline{A}B\overline{C}$	$\overline{A}BC$
$A\overline{B}$	$A\overline{B}\,\overline{C}$	$A\overline{B}C$
AB	$AB\overline{C}$	ABC

	$\overline{C}$	C
$\overline{A}\,\overline{B}$	0	0
$\overline{A}B$		
$A\overline{B}$	0	
AB	0	0



	$\overline{C}$	C
$\overline{A}\,\overline{B}$		
$\overline{A}B$	0	0
$A\overline{B}$		0
AB		



Layout of a 3-input K-map based on Gray Code

$\overline{A}B+\overline{A}BC+ABC$

$\overline{A}B+BC(\overline{A}+A)$

$\overline{A}B+BC(1)$

$\overline{A}B+BC$

$\overline{A}\,\overline{B}+A\overline{B}+AB\overline{C}+A\overline{B}\,\overline{C}$

$\overline{B}(\overline{A}+A)+A\overline{C}(B+\overline{B})$

$\overline{B}(1)+A\overline{C}(1)$

$\overline{B}+A\overline{C}$

K-map: 4-input example

$$BD + C\bar{D}$$

K-Map Minimization Guideline

- Loop all isolated 1s;
- Consider each remaining 1 separately. If it can be looped in more than one way, try include it in the largest possible loop;
- A minimal solution is derived as soon as all 1s are covered. In the process of making the largest loop, it is permissible to use previously covered 1s.

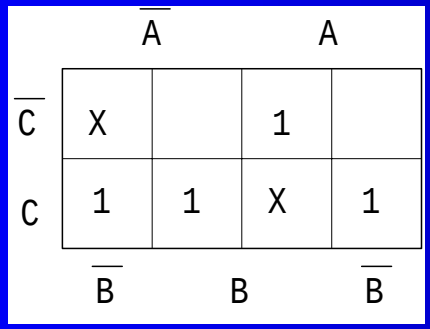
Class Exercise

Simplify the following logic function:

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	0	1
$\bar{A}B$	0	0	0	0
AB	1	0	0	1
$A\bar{B}$	1	0	0	1

Don't Care Condition in K-Map

Certain combinations of inputs may be immaterial to a given function. For such don't care states the output is irrelevant and may be 1 or 0.



X – don't care input condition

If we use X=0, $F = \overline{A}\overline{B}C + \overline{A}BC + A\overline{B}C + AB\overline{C}$

This can be simplified (with X=0) to $F = \overline{A}C + A\overline{B}C + AB\overline{C}$

With K-map, 1 can be assigned to any don't care position to

- form largest possible loop
- Combine isolated 1 to form a loop
- In the example we can further simplify the expression by taking the bottom X as 1: $F = C + AB$

Design Example: Majority Detector

Design a 4-input circuit that will function as a majority detector. The circuit should output high when a majority of the inputs are high.

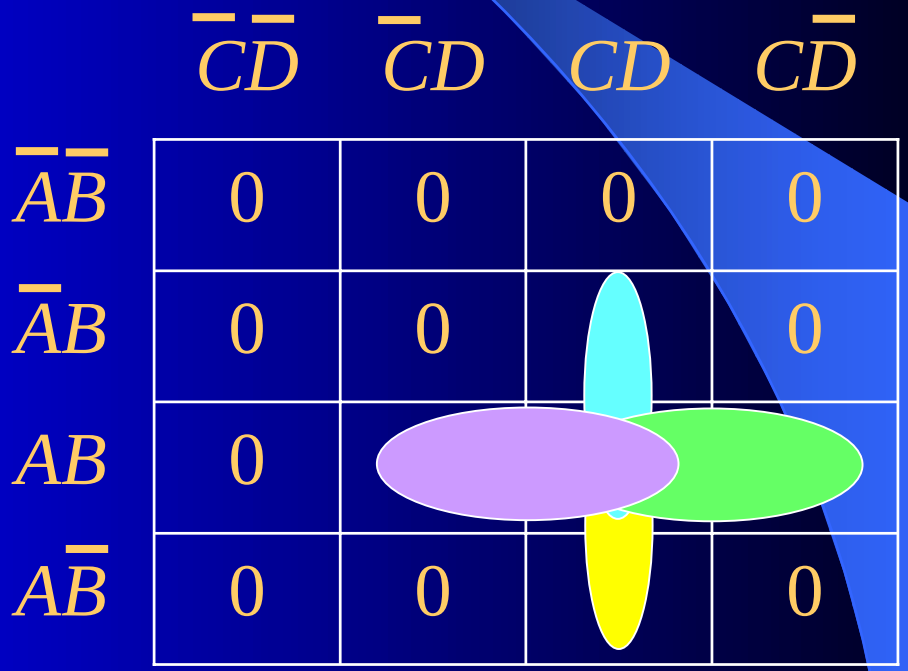
The first step is to complete a truth table and mark high outputs for every set of input conditions that contains three or four (majority) 1s. This is shown in the truth table on the right.

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

Design Example: Majority Detector

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

Next, plot the Boolean expression from the truth table on a K-map as shown and simplify the expression.



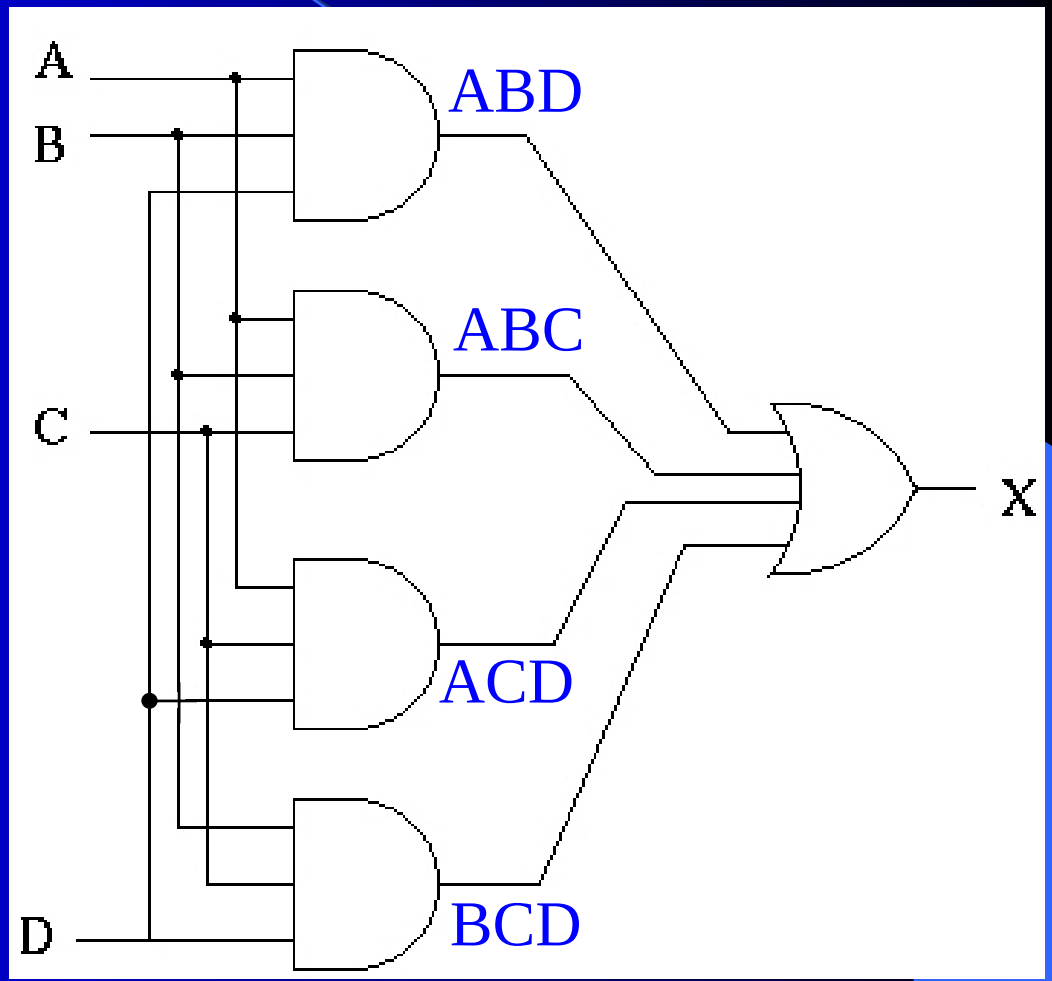
$$X = ABD + BCD + ABC + ACD$$

Design Example: Majority Detector

The simplified expression is shown on the right, which has four terms since four pairs of 1s can be looped on the K-map.

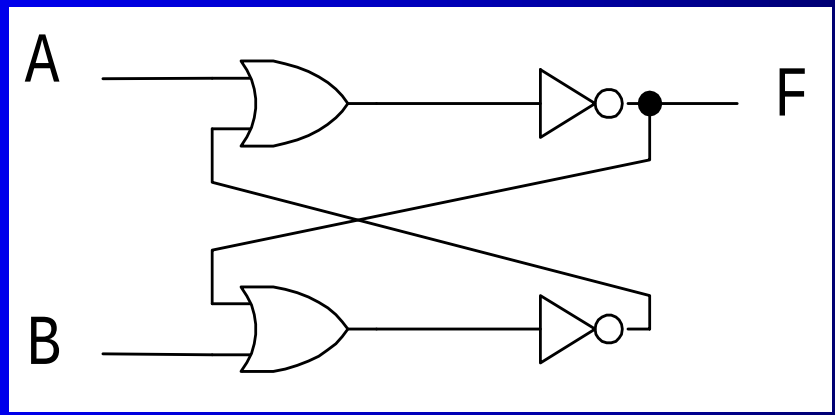
Implementation is straightforward as shown on the right because this is an SOP expression. The circuit requires four 3-input AND gates and one 4-input OR gate.

$$X = ABD + BCD + ABC + ACD$$



Types of Digital Networks

- Combinational: The logic outputs at each instant are determined only by the inputs at that instant
- Sequential: The logic output are dependent on the previous inputs as well as the present inputs



When A=B=0, F may be either 0 or 1 depending on the previous input.

Common Combinational Logic Circuits

Encoder

A circuit converts a signal to a coded format.

Example: Decimal to Binary Coded Decimal (BCD) encoder

Decimal signal lines	BCD $b_3b_2b_1b_0$
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

BCD Invalid Sum Detector

Decimal signal lines	BCD $b_3b_2b_1b_0$
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

Decimal signal lines	Invalid BCD $b_3b_2b_1b_0$
-	1 0 1 0
-	1 0 1 1
-	1 1 0 0
-	1 1 0 1
-	1 1 1 0
-	1 1 1 1

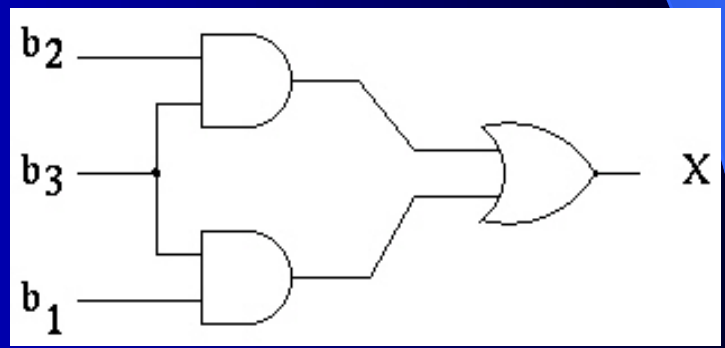
Invalid BCD numbers

BCD Invalid Sum Detector

Invalid BCD $b_3b_2b_1b_0$
1 0 1 0
1 0 1 1
1 1 0 0
1 1 0 1
1 1 1 0
1 1 1 1

	$\bar{b}_3\bar{b}_2$	$\bar{b}_3b_2$	b_3b_2	$b_3\bar{b}_2$
$\bar{b}_1\bar{b}_0$	0	0		0
$\bar{b}_1b_0$	0	0		0
b_1b_0	0	0		
$b_1\bar{b}_0$	0	0		

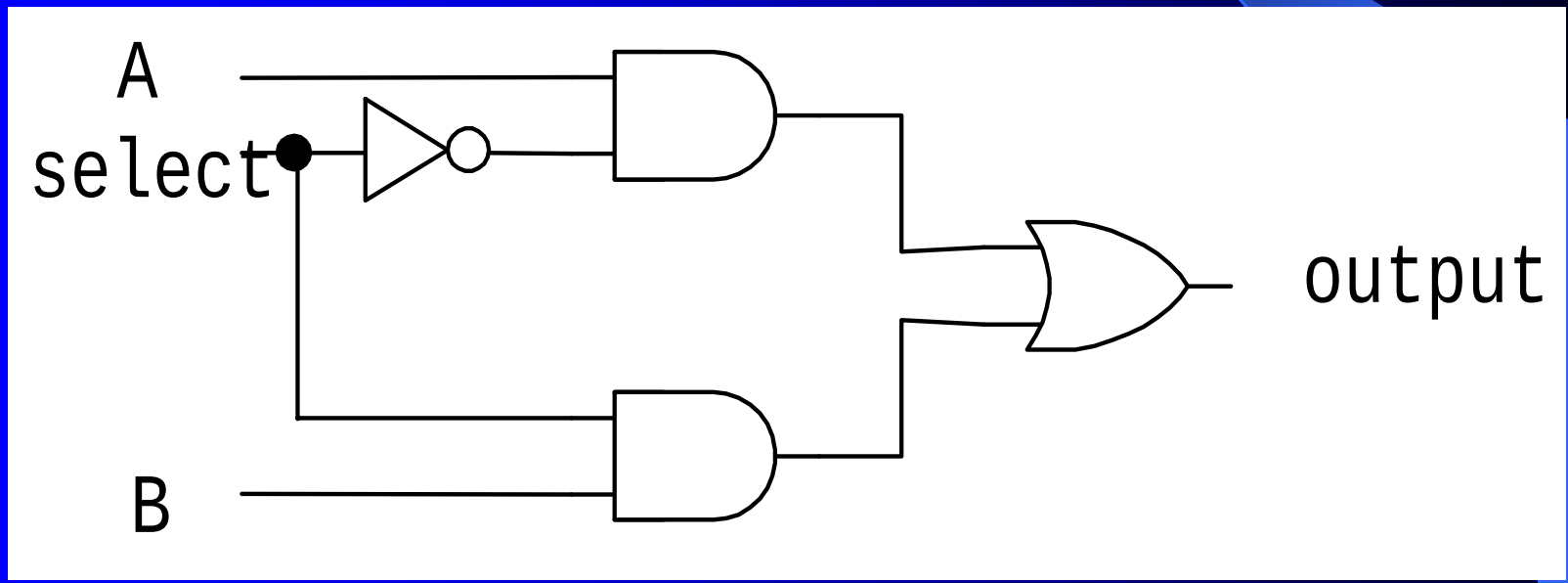
$$X = b_3b_2 + b_3b_1$$



Multiplexer

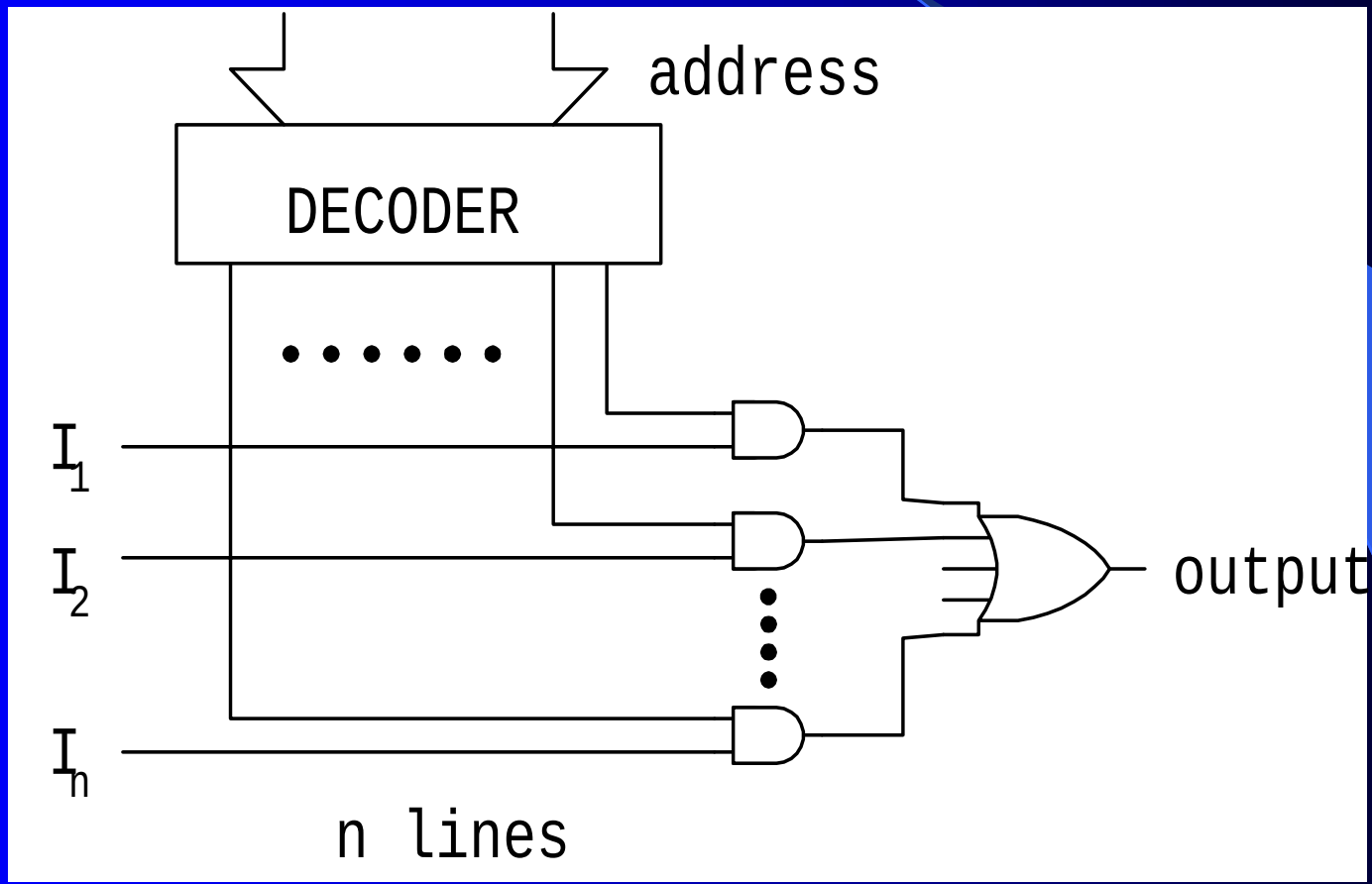
Select one signal from two or more input lines and transmit it on a single output line.

Example: 2-1 multiplexer



Multiplexer

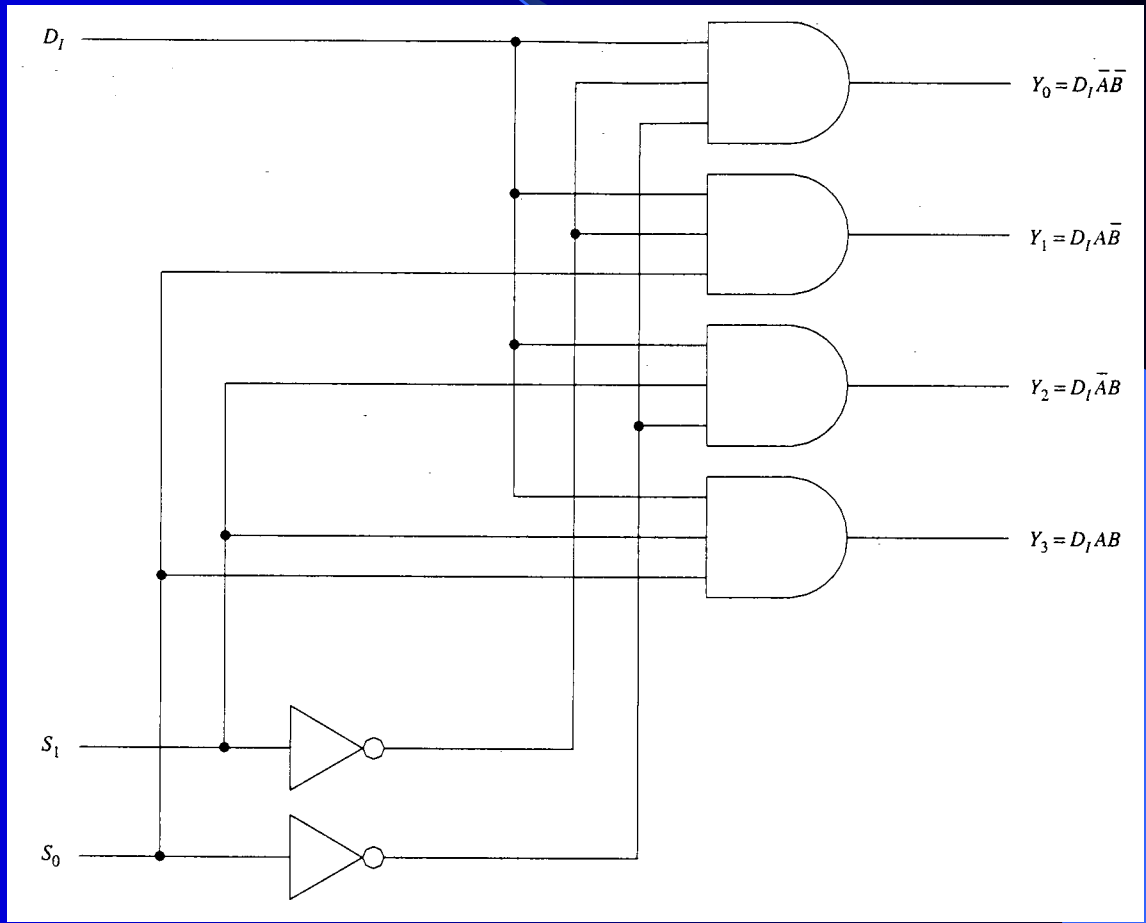
Example: n-1 multiplexer



Demultiplexer

Recover multiplexed signals and transmit them to separate outputs.

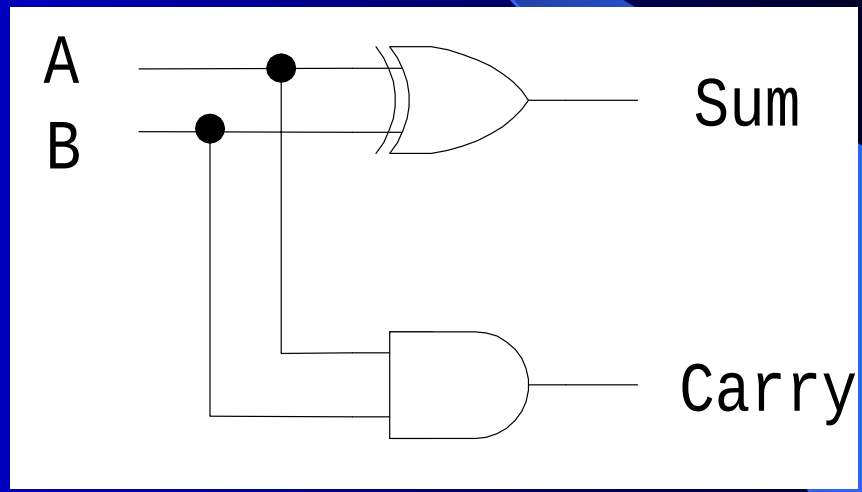
e.g. 4-channel demultiplexer



Half Adder

Add 2 bits and produce both a sum and a carry output

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Full Adder

Sum of three bits: 2 data bits and a carry bit from lower bit addition

A	B	C	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	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

Sum

	$\overline{A}$		A	
$\overline{C}$	0	1	0	1
C	1	0	1	0
	$\overline{B}$	B		$\overline{B}$

$$SUM = \overline{A}\overline{B}C + \overline{A}B\overline{C} + ABC + A\overline{B}\overline{C}$$

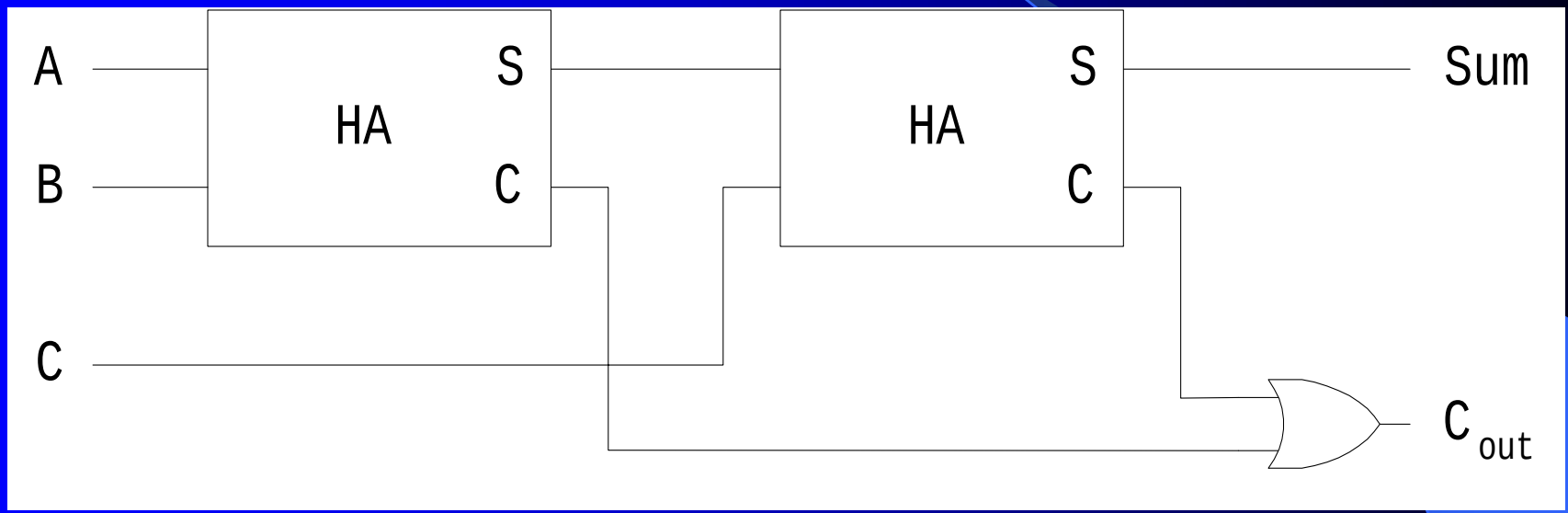
C_{out}

	$\overline{A}$		A	
$\overline{C}$	0	0	1	0
C	0	1	1	1
	$\overline{B}$	B		$\overline{B}$

$$C_{OUT} = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

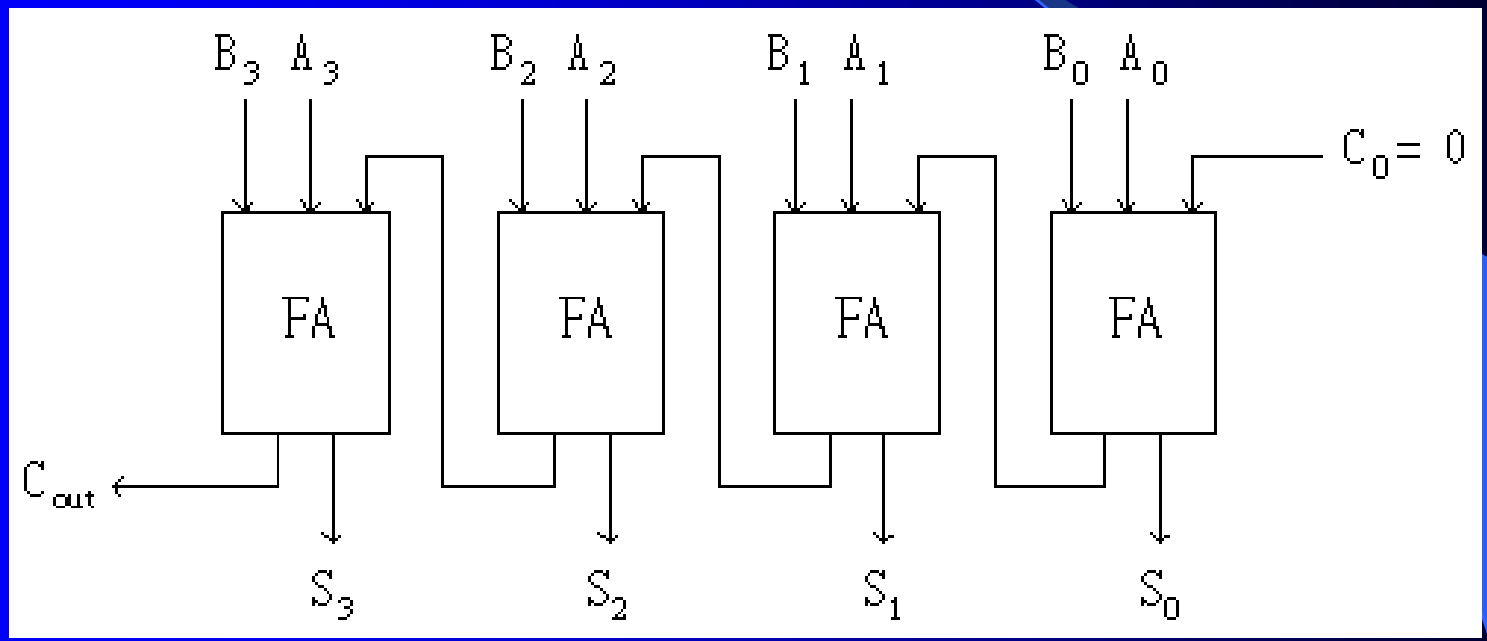
$$C_{OUT} = AB + BC + AC$$

Building Full Adder from Half Adder



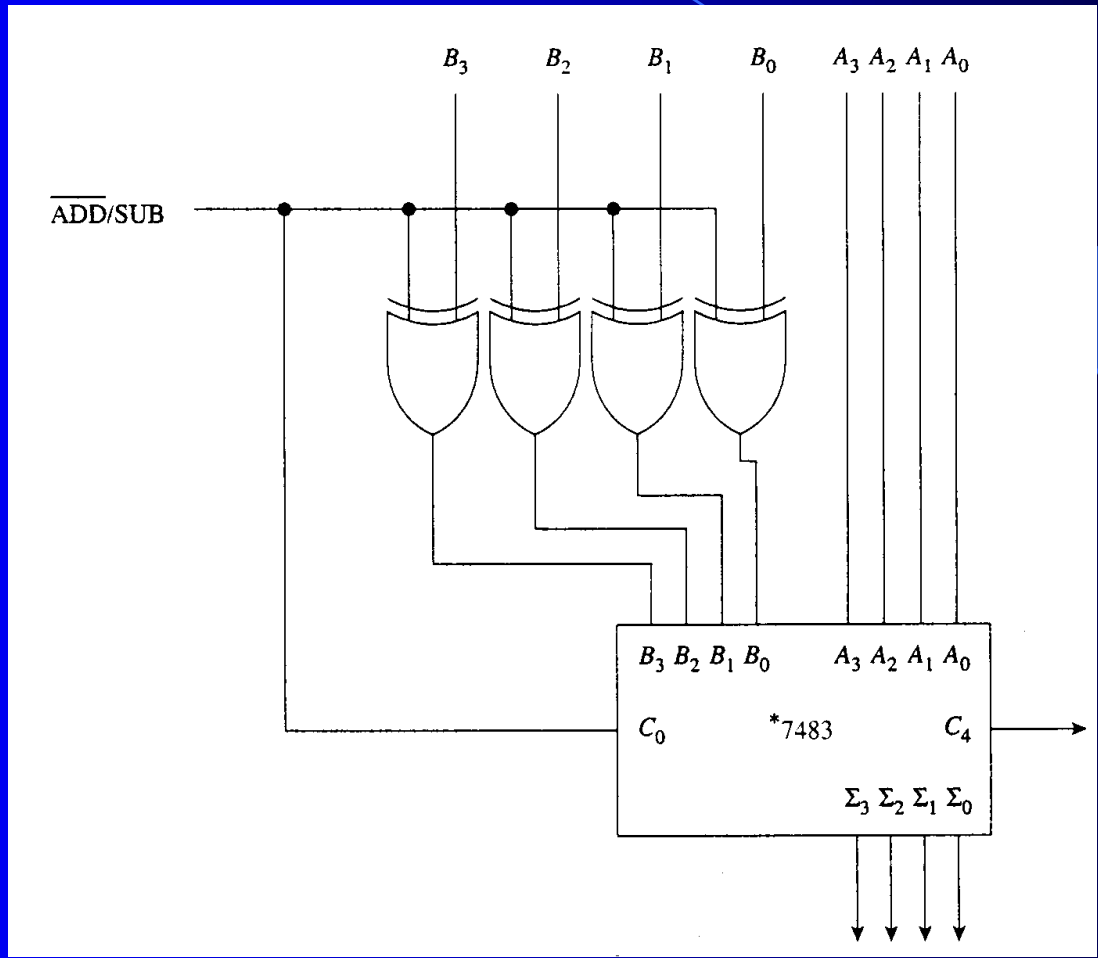
Multi-bit adder circuits

Parallel Adder



Number of modules same as the word length

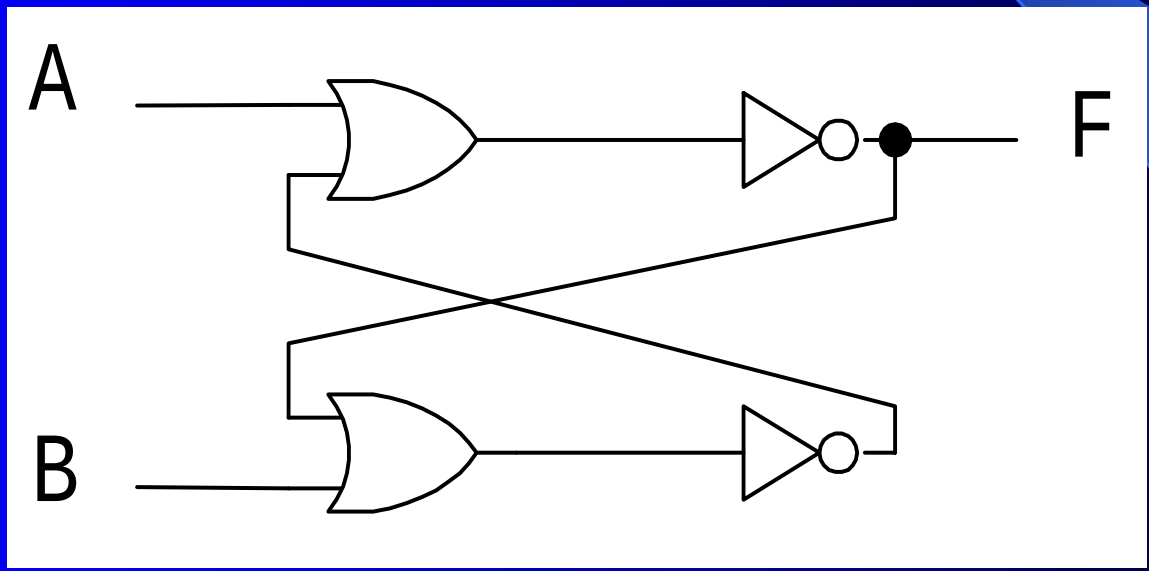
Adder/Subtractor



Sequential Logic Circuits

Sequential Logic Circuits

A sequential logic circuit's output depends on its previous state (condition) in addition to its current inputs. This is accomplished by using feedback from the circuit's outputs back to its inputs.



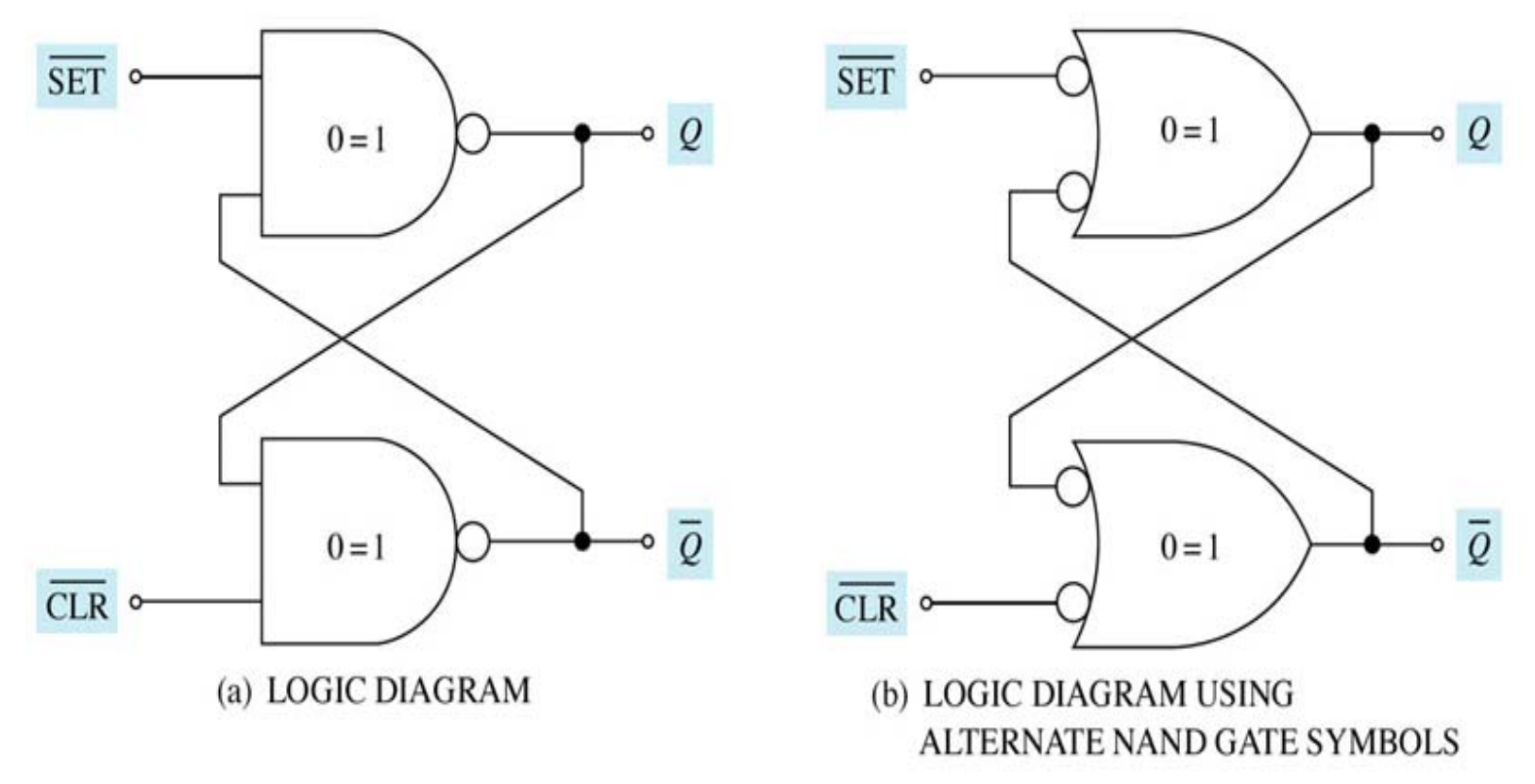
When A=B=0, F may be either 0 or 1 depending on the previous input.

Latch

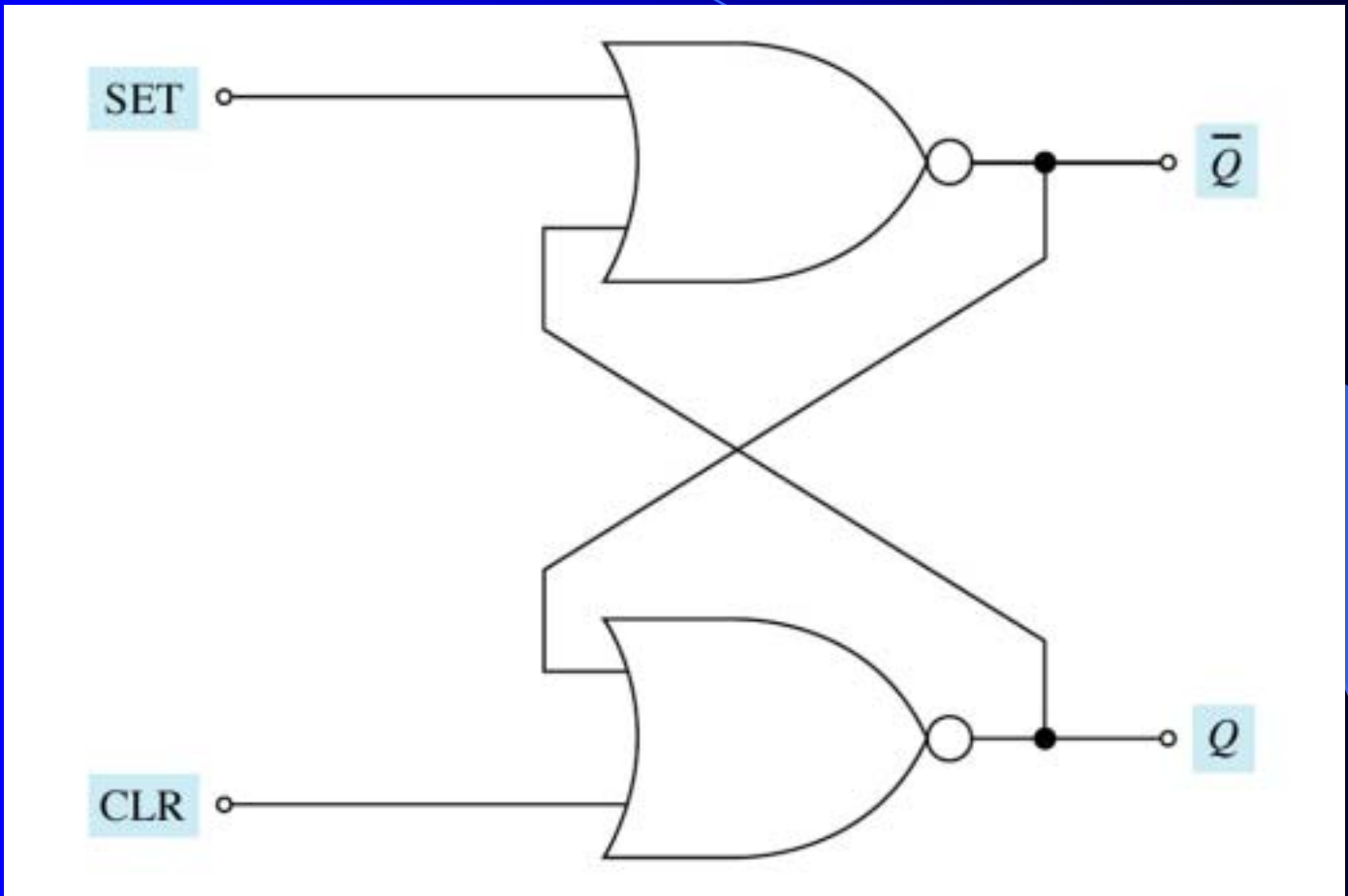
A latch circuit is a bistable device. *Bistable* indicates that the latch has two stable states. These two latch states are called the SET state and the CLEAR state. Once a latch is put in one of these states it will remain in that state until forced to change states by another input signal.

There are two basic types of latch circuits: the NAND-gate latch and the NOR-gate latch.

NAND-gate Latch



NOR-gate Latch



Latches

These latches are formed by using cross-coupled inverting logic gates. The cross coupling provides the feedback necessary for the latch circuit to retain (store) data. A latch constructed with NAND gates is referred to as an active-low latch, and a latch constructed with NOR gates is called an active-high latch. The active-low and active-high references are derived from the input logic level that must be applied to the latch to put it in a certain state.

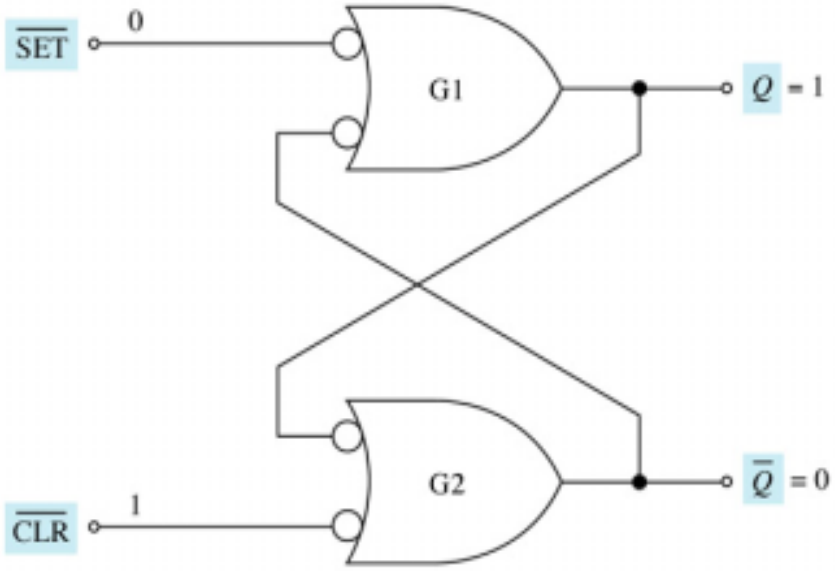
Latches

A latch has two outputs. One is labeled Q. The other output, which is the complement of Q, is labeled $\overline{Q}$. A latch can have only two valid output conditions. One is the SET state, where output $Q = 1$ and $\overline{Q} = 0$. The other condition is the CLEAR state, where $Q = 0$ and $\overline{Q} = 1$. Since the latch is designed to normally have complementary outputs (Q and $\overline{Q}$), it is only necessary to remember that Q is high in the SET state and Q is low in the CLEAR state.

Latches

$\overline{Q}$ of course, will normally be the opposite level of Q, The $\overline{Q}$ output is a convenience for circuit designers, is often not used in digital circuits, and is sometimes not available as an output on a flip-flop IC. The CLEAR state is also referred to as the RESET state. The latches are sometimes called S-C (SET-CLEAR) latches or S-R (SET-RESET) latches. Since a latch only has a SET state or a CLEAR state, it can store only *one bit* of data. Latch circuits are typically used to store binary information on a temporary basis.

State Table – SET State



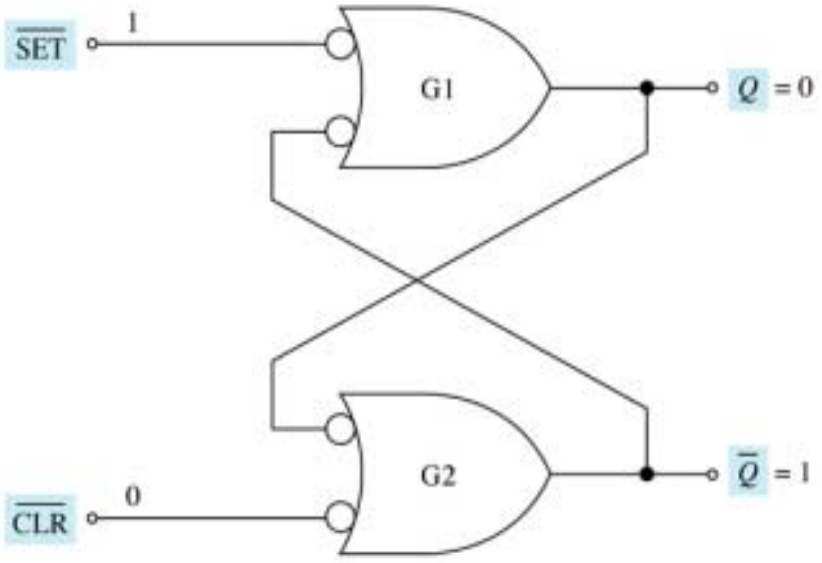
(a) LOGIC DIAGRAM

$\overline{SET}$	$\overline{CLR}$	Q	$\overline{Q}$	STATE
0	1	1	0	SET

(b) STATE TABLE

A latch must be put in a SET state to store a binary 1 data bit at the *Q* output.

State Table – CLEAR State



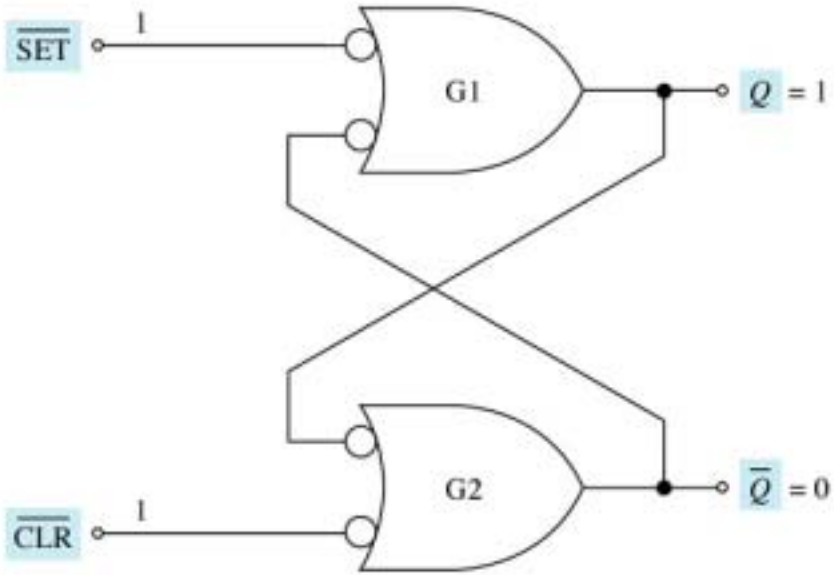
(a) LOGIC DIAGRAM

$\overline{SET}$	$\overline{CLR}$	Q	$\overline{Q}$	STATE
1	0	0	1	CLEAR

(b) STATE TABLE

A latch must be put in a CLEAR state to store a binary 0 at the *Q* output.

State Table – RETAIN State



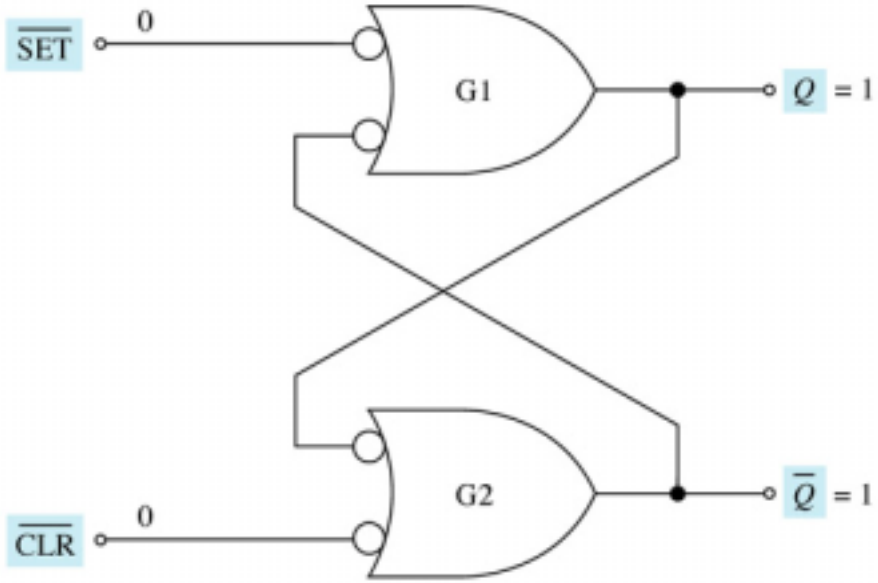
(a) LOGIC DIAGRAM

$\overline{SET}$	$\overline{CLR}$	Q	$\overline{Q}$	STATE
1	1	1 / 0	0 / 1	RETAIN

(b) STATE TABLE

The two logic 1s applied to this latch cause it to retain the condition it was in when the two inputs were brought high (inactive).

State Table – INVALID State



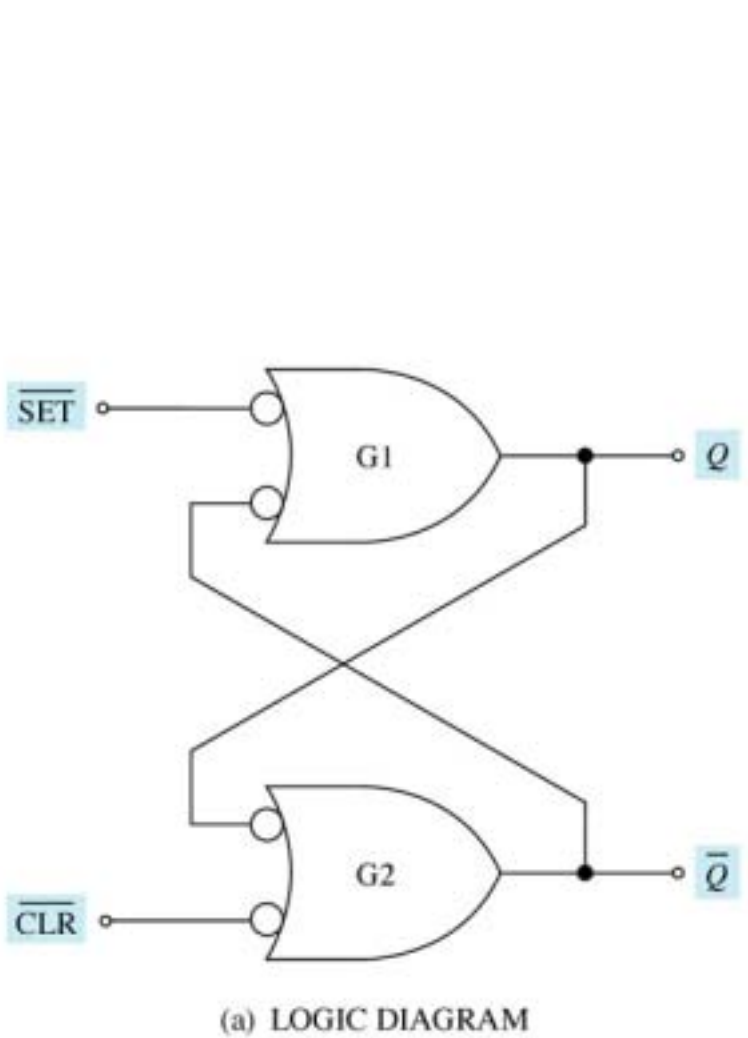
(a) LOGIC DIAGRAM

$\overline{SET}$	$\overline{CLR}$	Q	$\overline{Q}$	STATE
0	0	1	1	INVALID

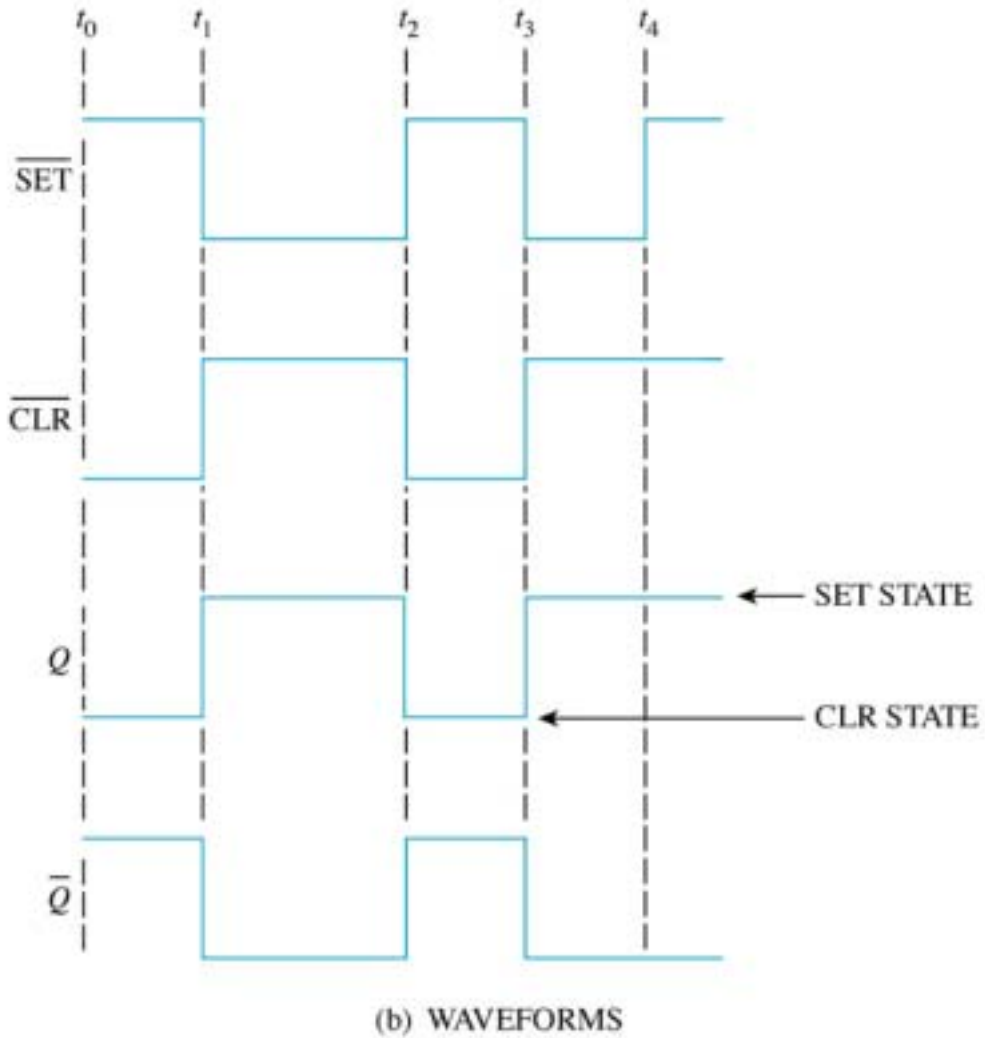
(b) STATE TABLE

For a NAND gate, 0s into each logic gate will produce a 1 out of each gate. Equal outputs are considered INVALID.

Active-Low Latch Operation

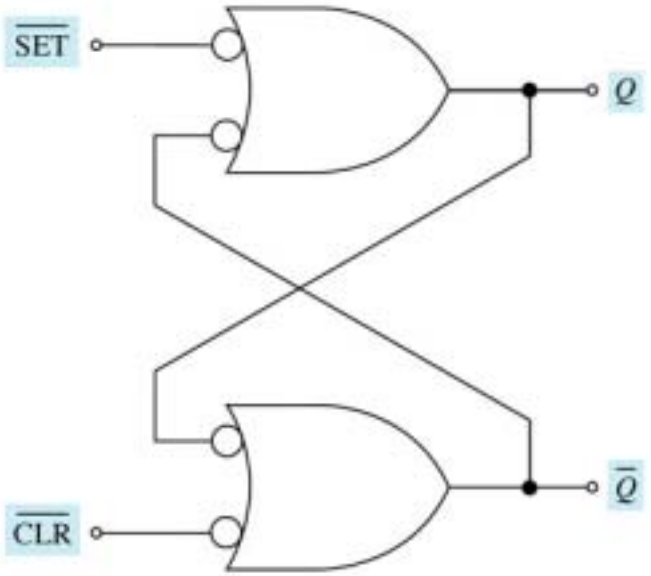


(a) LOGIC DIAGRAM



(b) WAVEFORMS

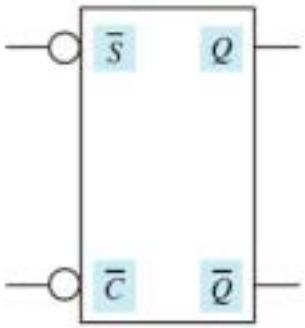
Active-Low Latch



(a) LOGIC DIAGRAM

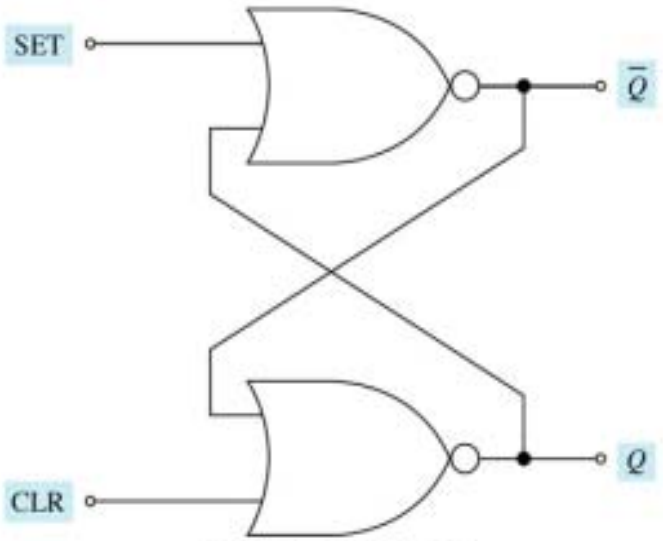
$\overline{SET}$	$\overline{CLR}$	Q	$\overline{Q}$	STATE
0	1	1	0	SET
1	0	0	1	CLEAR
1	1	1 / 0	0 / 1	RETAIN
0	0	1	1	INVALID

(b) STATE TABLE



(c) LOGIC SYMBOL

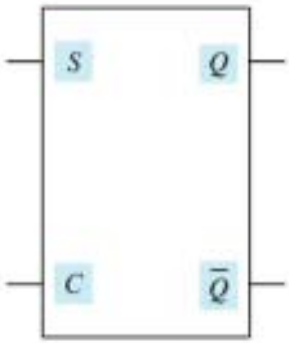
Active-High Latch



(a) LOGIC DIAGRAM

SET	CLR	Q	$\overline{Q}$	STATE
1	0	1	0	SET
0	1	0	1	CLEAR
0	0	1 / 0	0 / 1	RETAIN
1	1	0	0	INVALID

(b) STATE TABLE



(c) LOGIC SYMBOL