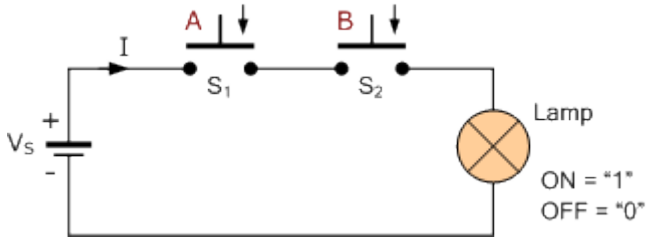
Contents

2. Switch and lamp logic

Condition	Switch	Comment
1	Open	No light produced =logic 0
2	Closed	Light produced = logic 1

Consider the circuit with two switches shown in Fig. 10.3. Here the lamp will only operate when switch A is closed and switch B is closed.Let's look at the operation of the circuit. Since there are two switches (A and B) and there are two possible states for each switch (open or closed), there is a total of four possible conditions for the circuit.



AND logic

Condition	Switch A	Switch B	Comment
1	Open	Open	No light produced =logic 0
2	Open	closed	No light produced =logic 0
3	closed	Open	No light produced =logic 0
4	Closed	closed	Light produced = logic 1

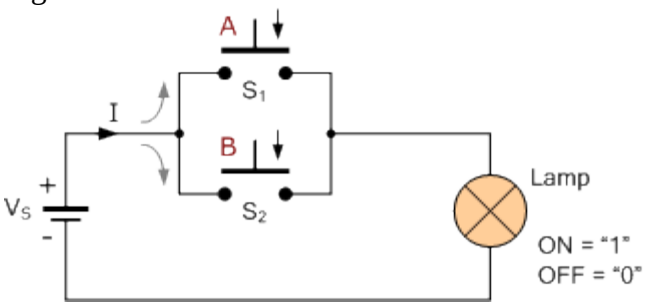
We can summarize these conditions in Table

Since each switch can only be in one of the two states (i.e. open or closed) at any given time, the open and closed conditions are mutually exclusive. Thus represent the logical states of the two switches by the binary digits, 0 and 1. Once again, if we adopt the convention that an open switch can be represented by 0 and a closed switch by 1, we can rewrite the truth table in terms of the binary states shown in Fig. 10.4 where: No light (off) = 0 Light (on) = 1

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

OR logic

Figure 10.5 shows another circuit with two switches.



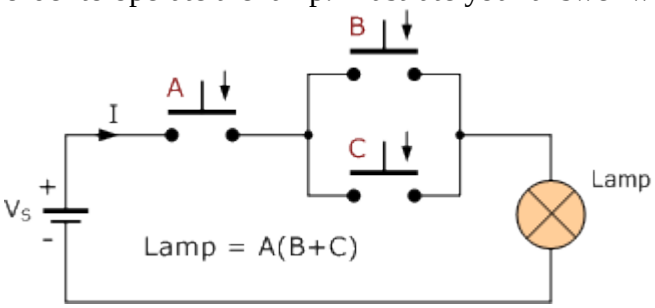
In this case the lamp will operate when either of the two switches is closed. As before, there is a total of four possible conditions for the circuit. We can summarize these conditions in Table 10.3.

Condition	Switch A	Switch B	Comment
1	Open	Open	No light produced =logic 0
2	Open	closed	light produced =logic 1
3	closed	Open	light produced =logic 1
4	Closed	closed	Light produced = logic 1

Once again, adopting the convention that an open switch can be represented by 0 and a closed switch by 1, we can rewrite the truth table in terms of the binary states as shown in Fig.

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

Numerical 1: Figure 10.7 shows a simple switching circuit. Describe the logical state of switches A, B, and C in order to operate the lamp. Illustrate your answer with a truth table.



Solution

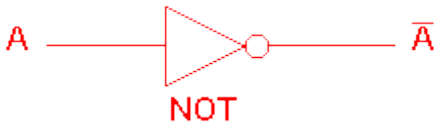
In order to operate the lamp, switch A and either switch B or switch C must be operated. The truth table is shown in Fig. 10.8.

A	B	C	Y=A(B+C)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

3. Logic gates

i) NOT gate

The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an *inverter*. If the input variable is A, the inverted output is known as NOT A. This is also shown as A', or A with a bar over the top, as shown at the outputs.

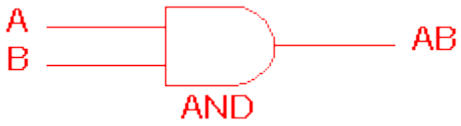


NOT gate	
A	$\overline{A}$
0	1
1	0

ii) OR gate: The OR gate is an electronic circuit that gives a high output (1) if **one or more** of its inputs are high. A plus (+) is used to show the OR operation.

2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

iii) AND gate: The AND gate is an electronic circuit that gives a **high** output (1) only if **all** its inputs are high. A dot (.) is used to show the AND operation i.e.Y= A.B.



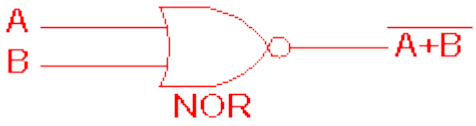
2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

iv) NAND gates

NAND (i.e. NOT-AND) gates will only produce a logic 0 output when all inputs are simultaneously at logic 1. Any other input combination will produce a logic 1 output. The circle shown at the output denotes this inversion.

2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

v) NOR gate: This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate. The outputs of all NOR gates are low if **any** of the inputs are high. The symbol is an OR gate with a small circle on the output. The small circle represents inversion



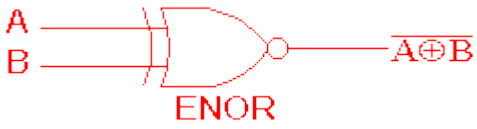
2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

.

vi) EX-OR gate: The '**Exclusive-OR**' gate is a circuit which will give a high output if **either, but not both**, of its two inputs are high. An encircled plus sign (⊕) is used to show the EOR operation.

2 Input EXOR gate		
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

vii) EX-NOR gate: The '**Exclusive-NOR**' gate circuit does the opposite to the EOR gate. It will give a low output if **either, but not both**, of its two inputs are high. The symbol is an EXOR gate with a small circle on the output. The small circle represents inversion



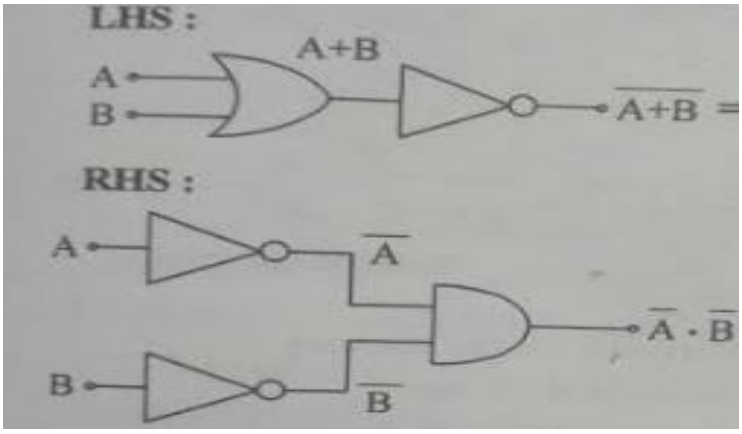
2 Input EXNOR gate		
A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

4. DeMorgans Theorem (state & prove).

De Morgan's First Theorem:-

Statement - The complement of a logical sum equals the logical product of the complements.

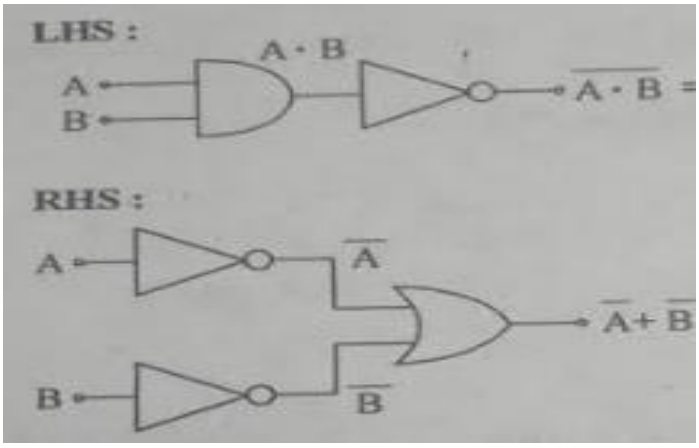
Logic equation $\overline{A + B} = \bar{A}.\bar{B}$



De Morgan's Second Theorem:-

Statement - The complement of a logical product equals the logical sum of the individual complements.

Logic equation $\overline{A.B} = \bar{A} + \bar{B}$



Truth Table to prove De Morgan's Theorem

A	B	$\overline{A}$	$\overline{B}$	A+B	$A \cdot B$	$\overline{A+B}$	$\overline{A} \cdot \overline{B}$	$\overline{A \cdot B}$	$\bar{A} + \bar{B}$
0	0	1	1	0	0	1	1	1	1
0	1	1	1	1	0	0	0	1	1
1	0	0	0	1	0	0	0	1	1
1	1	0	0	1	1	0	0	0	0

5. Rules & Laws of Boolean Algebra (12 rules & 3 Laws).

Basic Rules of Boolean Algebra

1. $A + 0 = A$	7. $A \cdot A = A$
2. $A + 1 = 1$	8. $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	9. $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	10. $A + AB = A$
5. $A + A = A$	11. $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	12. $(A + B)(A + C) = A + BC$

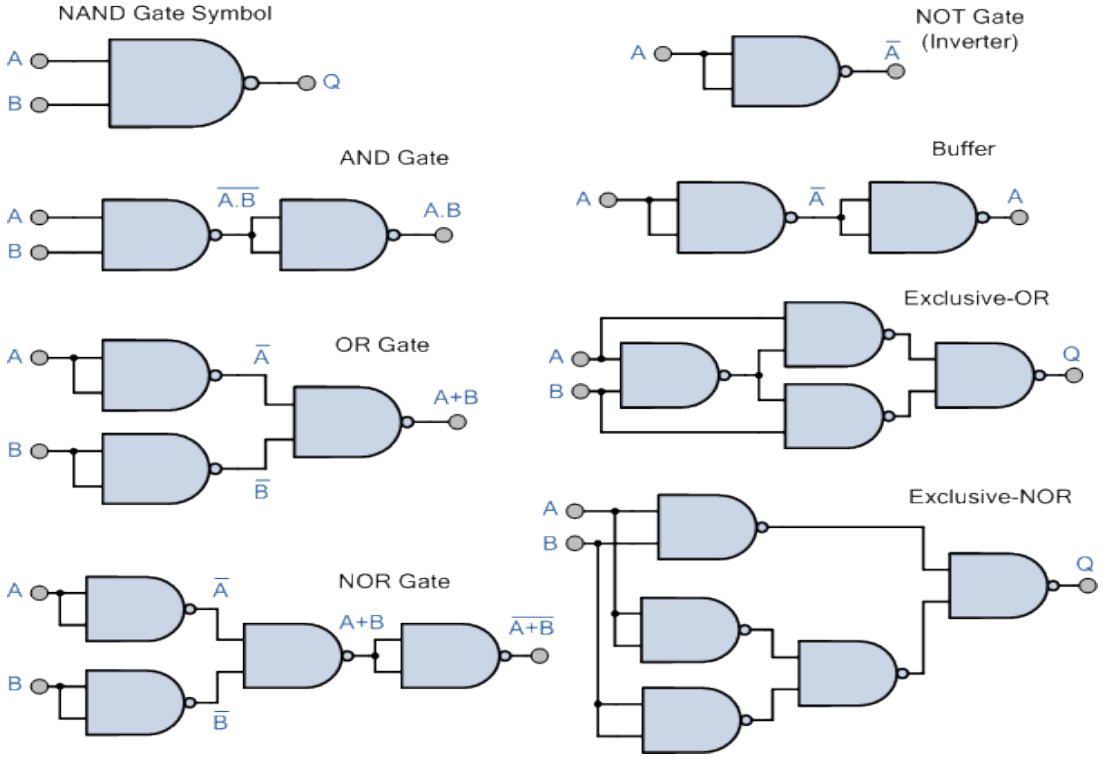
DeMorgan's Theorem

$\overline{(AB)} = (\overline{A} + \overline{B})$ $\qquad$ $\overline{(A + B)} = (\overline{A}\ \overline{B})$

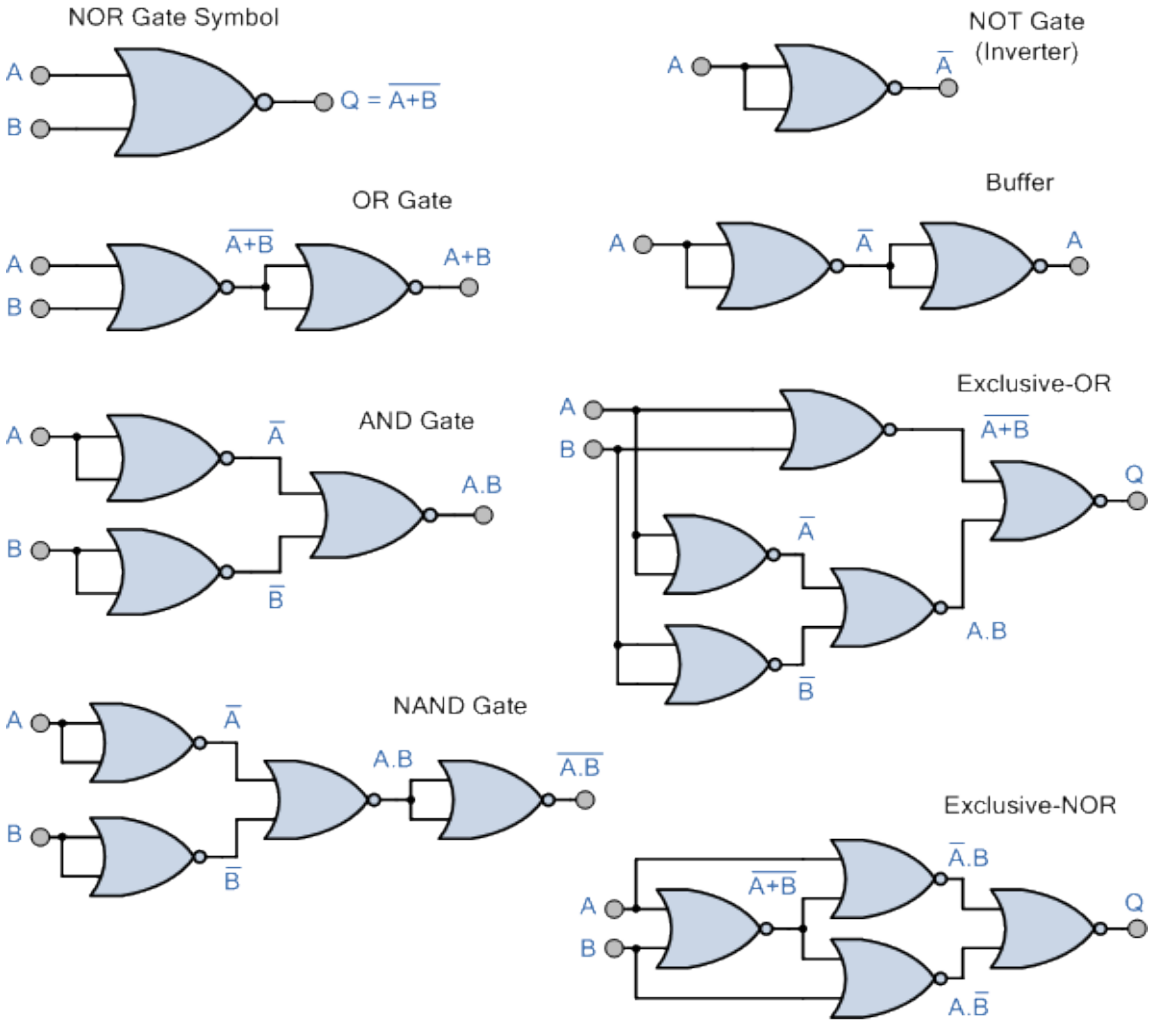
Reference	Rule or Law	Reference	Rule or Law
Rule 1	$A + 0 = A$	Rule 10	$A + AB = A$
Rule 2	$A + 1 = 1$	Rule 11	$A + A'B = A$
Rule 3	$A \bullet 0 = 0$	Rule 12	$(A + B)(A + C) = A + BC$
Rule 4	$A \bullet 1 = A$	Commutative Law	$A + B = B + A$ $AB = BA$
Rule 5	$A + A = A$		
Rule 6	$A + A' = 1$	Associative Law	$A + (B + C) = (A + B) + C$ $A(BC) = (AB)C$
Rule 7	$A \bullet A = A$		
Rule 8	$A \bullet A' = 0$	Distributive Law	$A(B + C) = AB + AC$
Rule 9	$(A')' = A$		

6. **The NAND and NOR gates are called *universal functions* since with either one the AND and OR functions and NOT can be generated.**

Implementation of all the gates usin only NAND Gates

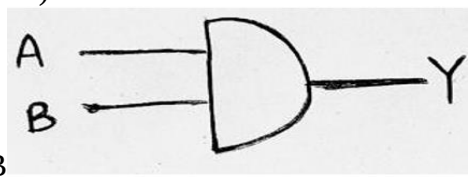


Implementation of all the gates using NOR Gates



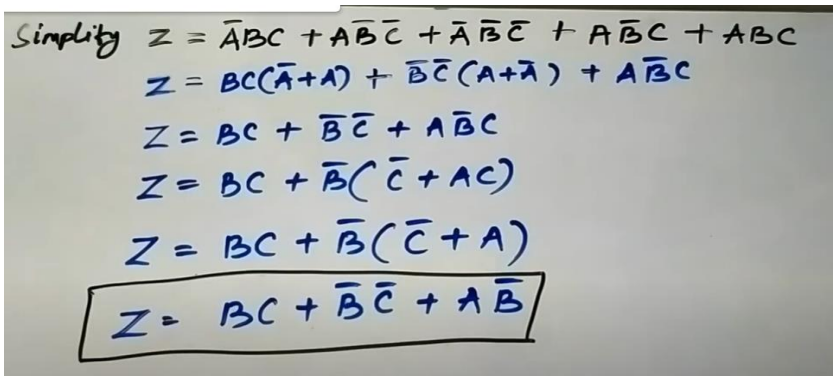
7. Simply & Realise Boolean functions using Basic gates

Y= (A+AB) . (B+BC) (C+AB)
= A (1+B) B(1+C) . (C+AB)
= A.1 .B.1 . (C+AB)
= AB (C+AB)
= (ABC+AB)
=A(B+BC)



Y = AB

2.



.

3. Y =AB+$\overline{AC}$ +A$\bar{B}$C(AB+C)

= AB+$\overline{AC}$ +A$\bar{B}$CAB+ A$\bar{B}$CC

= **AB**+$\overline{AC}$ +0+ **A$\bar{B}$C**

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

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

= AB+AC+$\overline{AC}$

= AB+1

=1

4. Simplify:Y = A'(A + B) + (B + AA)(A + B')

= A'A+A'B+ AB+AA+BB'+AB'

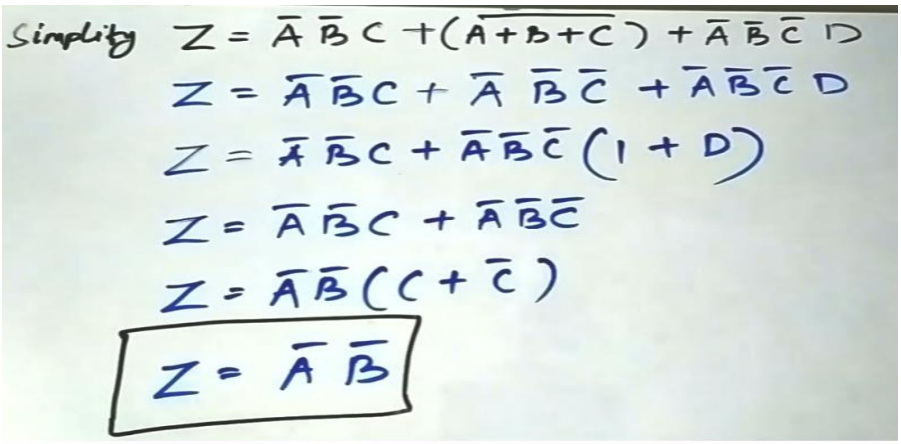
= 0+B(A'+A)+ A+0+AB'

= B.1+A(1+B')

= B+A.1

= A+B

5. $$(A+B)(A+C) = AA + AC + AB + BC$$
$$= A + AC + AB + BC$$
$$= A(1 + C + B) + BC$$
$$= A \cdot 1 + BC$$
$$= A + BC$$

6. 

7. $$A + \overline{A}\,B = A\,1 + \overline{A}\,B$$
$$= A\,(1 + B) + \overline{A}\,B$$
$$= A + A\,B + \overline{A}\,B$$
$$= A + B\,(A + \overline{A})$$
$$= A + B$$

TO WORK OUT

Simplify: F= (A + C)(AD + AD,) + AC + C

Ans: A+C

9. **Y =** $\overline{A\bar{B}}(\bar{A}$ **+ B) (**$\bar{B}$ **+B)**

10. **simplify** $(A' + B')\,(A + C') + B'\,(B + C)$

Ans: A'C'+B'

8. Combinational logic :

By using a standard range of logic levels (i.e. voltage levels used to represent the logic 1 and logic 0 states) logic circuits can be combined together in order to solve complex logic functions

Example 10.2 A logic circuit is to be constructed that will produce a logic 1 output whenever two, or more, of its three inputs are at logic 1.

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1➔BC
1	0	0	0
1	0	1	1➔ AC
1	1	0	1➔ AB
1	1	1	1➔ABC

Y= BC+AC+AB+ABC

Logic diagram

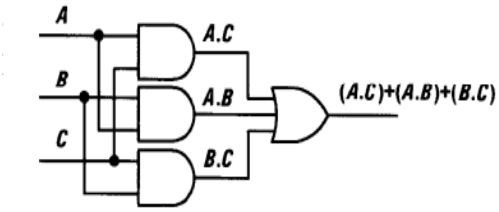


Figure 10.17 See Example 10.2

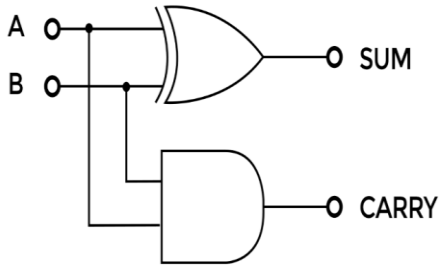
9. Adders: Half adder, Full adder, Full adder using two half adder

Half ADDER:

Half adder is the simplest of all adder circuits. Half adder is a combinational arithmetic circuit that adds two numbers and produces a sum bit (s) and carry bit (c) both as output. The addition of 2 bits is done using a combination circuit called a Half adder. The input variables are augend and addend bits and output variables are sum & carry bits. A and B are the two input bits.

Sum = A XOR B

Carry = A AND B

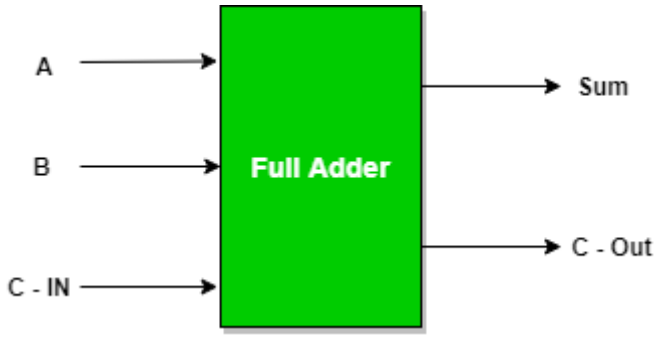


A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

FULL ADDER

Full Adder is the adder that adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM.

.

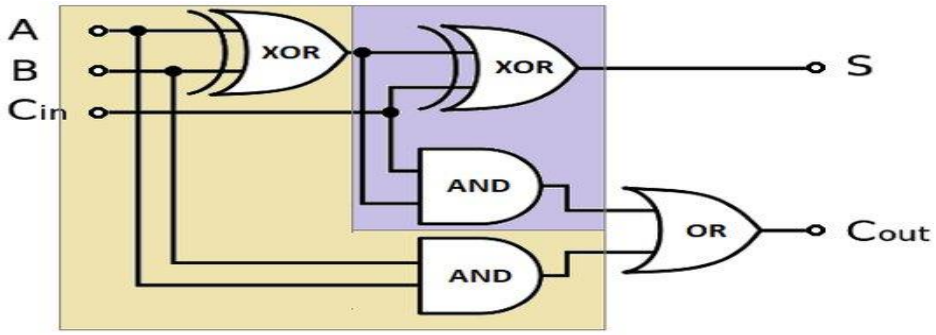


Inputs			Outputs	
A	B	C – IN	Sum	C - Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Logical Expression for
Logical Expression for SUM:
SUM = A’ B’ C-IN + A’ B C-IN’ + A B’ C-IN’ + A B C-IN
= C-IN (A’ B’ + A B) + C-IN’ (A’ B + A B’)
SUM = C-IN XOR (A XOR B)

Logical Expression for C-OUT:
C-OUT = A’ B C-IN + A B’ C-IN + A B C-IN’ + A B C-IN
= A B(C-IN’ + C-IN) + C-IN (AB’ + A’B)
= AB+ C-IN (A XOR B)

FULL ADDER USING 2 HALF ADDER



Truth table of the full adder

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
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

10. Sequential Logic: Introduction (Latch & FF & their differences using SR).

LATCH	FLIP – FLOP
Latches do not require clock signal.	Flip – flops have clock signals
A latch is an asynchronous device.	A flip – flop is a synchronous device.
Latches are transparent devices i.e. when they are enabled, the output changes immediately if the input changes.	A transition from low to high or high to low of the clock signal will cause the flip – flop to either change its output or retain it depending on the input signal.
A latch is a Level Sensitive device (Level Triggering is involved).	A flip – flop is an edge sensitive device (Edge Triggering is involved).
Latches are simpler to design as there is no clock signal (no careful routing of clock signal is required).	When compare to latches, flip – flops are more complex to design as they have clock signal and it has to be carefully routed. This is because all the flip – flops in a design should have a clock signal and the delay in the clock reaching each flip – flop must be minimum or negligible.
The operation of a latch is faster as they do not have to wait for any clock signal.	Flip - flops are comparatively slower than latches due to clock signal.
The power requirement of a latch is less.	Power requirement of a flip – flop is more.
A latch works based on the enable signal.	A flip – flop works based on the clock signal.

10.1 R-S bistables Latch

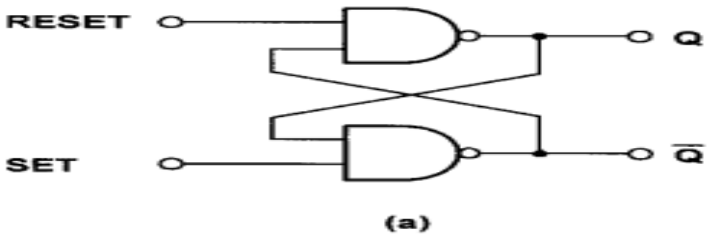
The simplest form of bistable is the R-S bistable. This device has two inputs, SET and RESET, and complementary outputs, Q and Q'.

Two simple forms of R-S bistable based on cross-coupled logic gates are shown in Fig. 10.19. Figure 10.19(a) is based on NAND gates while Fig. 10.19(b) is based on NOR gates.

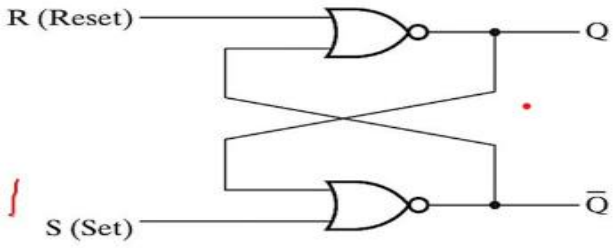
The simple cross-coupled logic gate bistable has a number of serious shortcomings (consider what Figure 10.19 R-S bistables using cross-coupled NAND and NOR gates would happen if a logic 1 was simultaneously present on both the SET and RESET inputs!) and practical forms of bistable make use of much improved purpose-designed logic circuits such as D-type and J-K bistables.

Case 1: S=1 R=0 will cause the Q output to become logic 1 while a logic 0 applied to the RESET input will cause the Q' output to become (or remain at) logic 0.

Case 2: S=0 R=1 will cause the Q output to become logic 0 while a logic 1 applied to the RESET input will cause the Q' output to become (or remain at) logic 1.



(a)



(a) Logic diagram

S	R	Q	$\overline{Q}$	
1	0	1	0	Set state
0	0	1	0	
0	1	0	1	Reset state
0	0	0	1	
1	1	0	0	Undefined

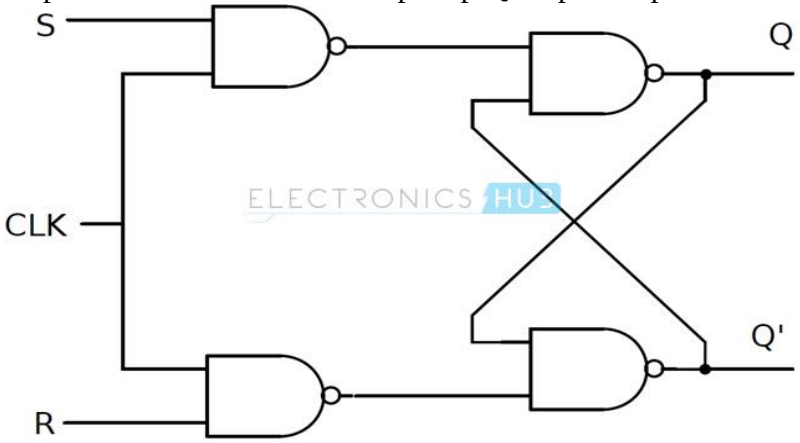
(b) Function table

10.2 SR Flip-flop

A Flip-Flop is a basic memory unit which can store 1-bit of digital information. It is a Bistable Electronic Circuit i.e.; it has two stable states: HIGH or LOW. As a flip-flop is a bistable element, its output remains in either of the stable states until an external event (known as a trigger) is applied.

In figure wiring or logical diagram of a NAND- gates clocked RS flip-flop, a clocked RS flip-flop consists of one basic flip-flop circuit and two additional NAND gates. As such, it has overall three inputs i.e. set input, reset input and clock (CLK) input or enabled input.

Here, **CLK input** behaves as an enabled signal for other two inputs (i.e. this flip-flop changes its condition only when a clock pulse is also applied along with an input variation). For example, when clock pulse or clock input is 1, flip-flop output tends to change. In other words, data existing on flip-flop input shifts on output. However, when clock input is zero, flip-flop output remains on its previous state no matter whatever the values of R and S inputs (i.e. no change reflects on output). In other words, when clock input or enable input (which is shortly written as EN) is low, no impact can be witnessed on flip-flop Q output in spite of changes in R and S. Th



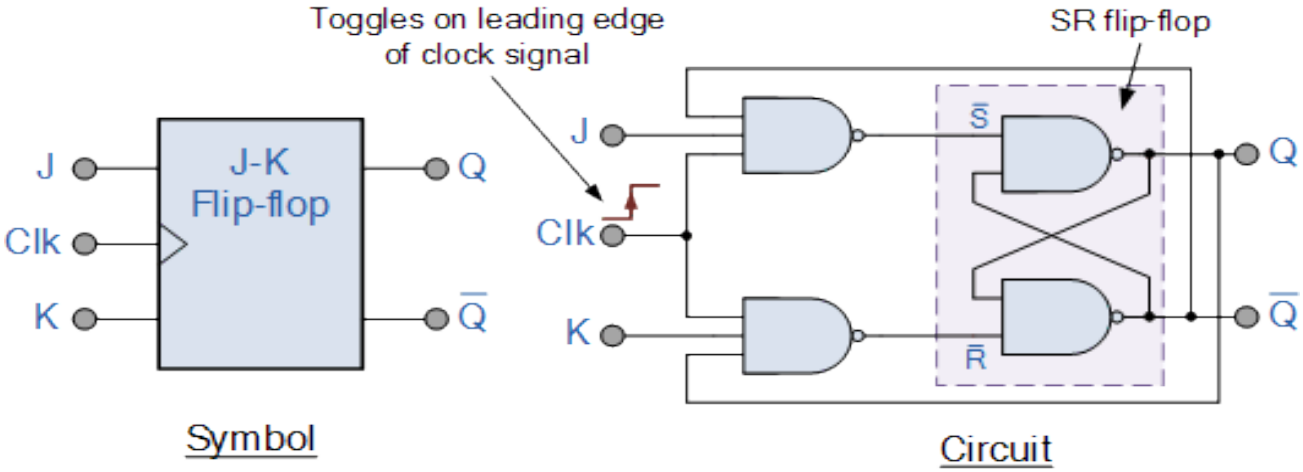
CLK	S	R	Output Q	State
↓ or 0 or 1	X	X	Last State	No Change (Hold)
↑	0	0	Last State	No Change (Hold)
↑	0	1	1	Set
↑	1	0	0	Reset
↑	1	1	Not Applied (?)	Forbidden

10.3 J K FlipFlop

The basic NAND gate RS flip-flop suffers from two main problems.

- Firstly, the condition when S = 0 and R = 0 should be avoided.
- Secondly, if the state of S or R changes its state while the input which is enabled is high, the correct latching action does not occur.

Thus to overcome these two problems of the RS Flip-Flop, the JK Flip Flop was designed.



J-K bistables have two clocked inputs (J and K), two direct inputs (PRESET and CLEAR), a CLOCK (CK) input, and outputs (Q and Q). Similarly, the PRESET and CLEAR inputs are invariably both active low (i.e. a 0 on the PRESET input will set the Q output to 1 whereas a 0 on the CLEAR input will set the Q output to 0). Tables 10.4 and 10.5 summarize the operation of a J-K bistable respectively for the PRESET and CLEAR inputs and for clocked operation

Truth Table

J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (toggles)

This table shows four useful modes of operation.

i) When J = K = 0 and clk = 1; output of both NAND gates will be 1; when any one input of NAND gate is 1 output of NAND gate will be complement of other input, so output remains as previous output or we can say the flip-flop is in the hold (or disabled) mode. In the hold mode, the data inputs have no effect on the outputs. The outputs "hold" the last data present.

ii) When J =0 K =1 and clk = 1; output of NAND gate connected to K will be Q' and corresponding NAND gate output will be 0; which RESETs the flip-flop.

iii) When J =1 K = 0 and clk = 1; output of NAND gate connected to J will be Q and corresponding NAND gate output will be 0; which the SETs the flip-flop.

iv) When J=1 K = 1 and clk = 1;, repeated clock pulses cause the output to turn off-on-off-on-off-on and so on.
This off-on action is like a toggle switch and is called toggling. Each clock pulse toggles the outputs to switch to their opposite states.

In the next clock pulse, the outputs will switch or "toggle" from set (Q=1 and Q'=0) to reset (Q=0 and Q'=1). Conversely, a "reset" state inhibits input K so that the flip-flop acts as if J=1 and K=0 when in fact both are 1. Then the next clock pulse toggles the circuit again from reset to set.

10.4 Four-stage binary counter using J-K bistables

Counters are one of the most useful parts of a digital system. A counter is a sequential circuit that holds the ability to count the number of clock pulses provided at its input.

Four-stage binary counter using J-K bistables

Four falling edge-triggered flip-flops are sequentially connected where the output of one flip-flop is provided as the input to the next. The input clock pulse is applied at the least significant or the first most flip-flop in the arrangement. Also, logic high signal i.e., 1 is provided at the J and K input terminals of the flip-flops.

Initially when the clock input is applied at the **LSB flip-flop** i.e.,

1. The output QA will change from 0 to 1 at the falling edge of the clock pulse. Further QA holds its state 1 and toggles from 1 to 0 only when another falling edge of the clock input is received. Again QA toggles from 0 to 1 at the next falling edge of the input clock pulse.

As we have already discussed that only the first flip-flop is triggered with an external clock signal.

2. So, now the output of flip-flop A will act as the clock input for flip-flop B and the external clock signal will not be going to affect QB.

3. Further for flip-flop C, the clock input will now be the output of flip-flop B i.e., QB. So, the output QC will be according to the transition of QB.As we can see in the diagram that first time QC toggles from 0 to 1 only at the first falling edge of QB signal. And maintains the state till it reaches the next falling edge of QB.
4. Further for flip-flop D, the clock input will now be the output of flip-flop C i.e., QC. So, the output QD will be according to the transition of QC. As we can see in the diagram that first time QD toggles from 0 to 1 only at the first falling edge of QC signal.

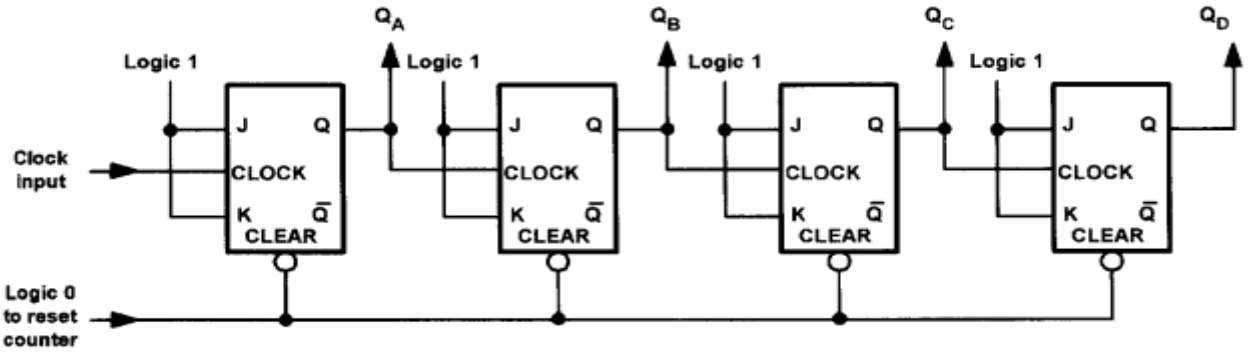


Figure 10.22 Four-stage binary counter using J-K bistables

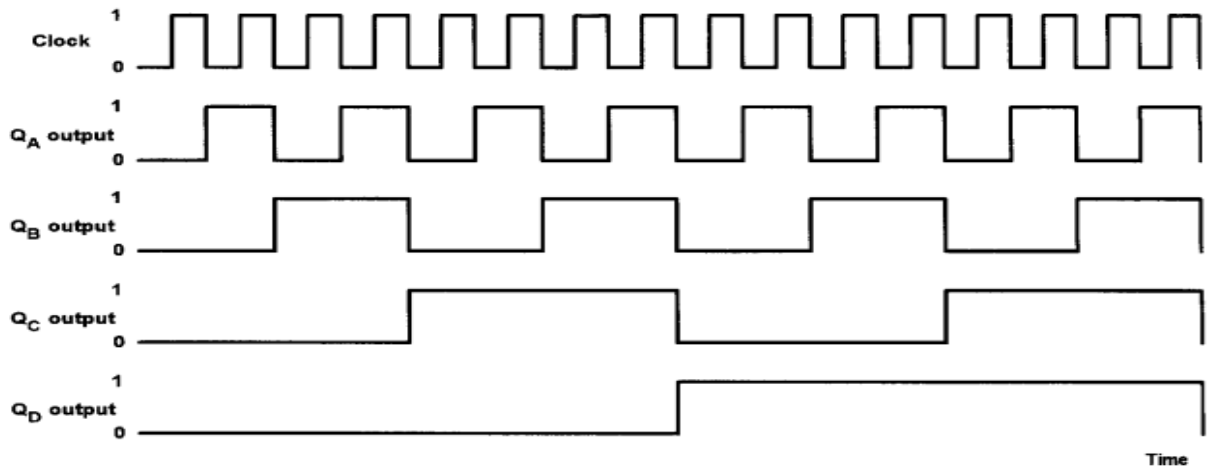
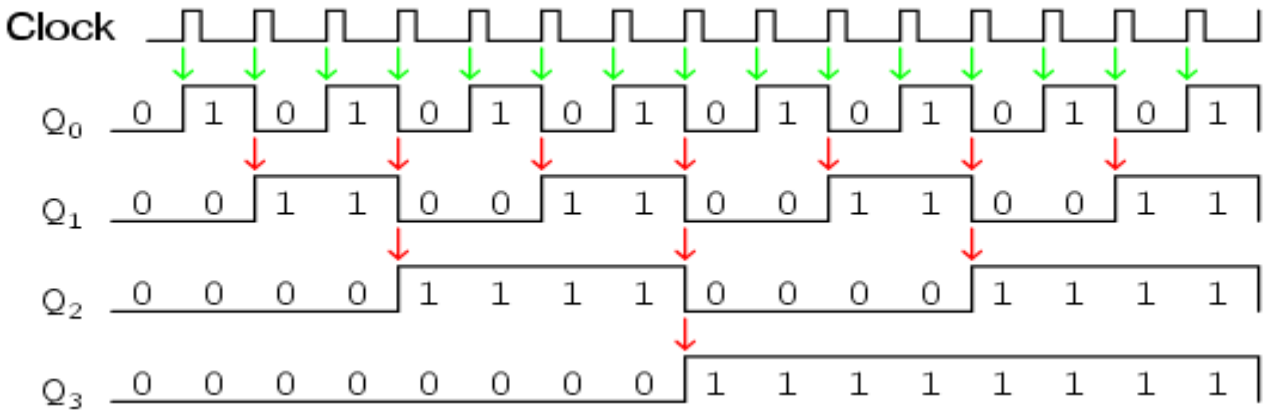


Figure 10.23 Timing diagram for the four-stage binary counter shown in Fig. 10.22

Figure 10.22 shows the arrangement of a four stage binary counter based on J-K bistables. The timing diagram for this circuit is shown in Fig. 10.23.
Each stage successively divides the clock input signal by a factor of two. Note that a logic 1 input is transferred to the respective Q-output on the falling edge of the clock pulse and all J and K inputs must be taken to logic 1 to enable binary counting.



MSB Q_3	Q_2	Q_1	LSB Q_0	Decimal Value
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15
0	0	0	0	full reset

Four Stage Shift Register Based On J-K Bistables.

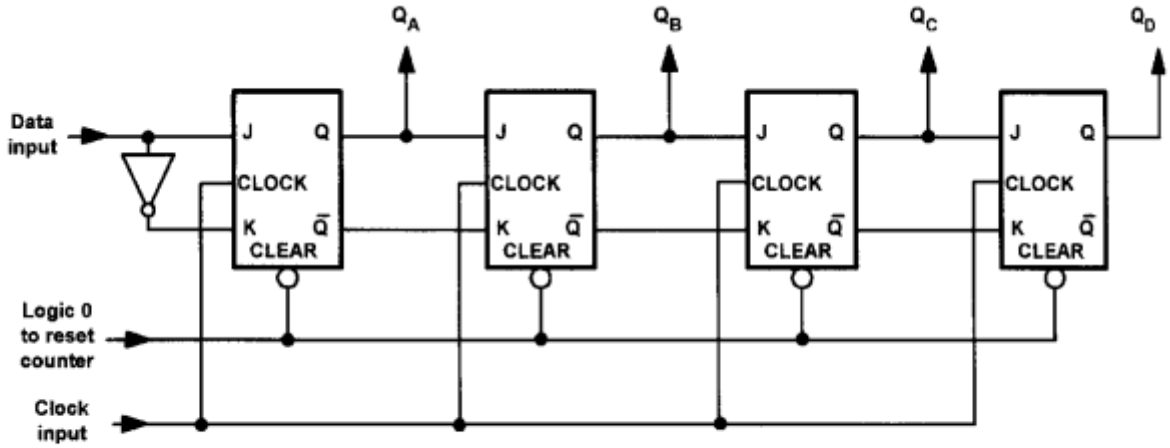


Figure 10.24 Four-stage shift register using J-K bistables

Figure 10.24 how the arrangement of a four stage shift register based on J-K bistables. The timing diagram for this circuit is shown in Fig. 10.25. Note that each stage successively feeds data to the next stage. Note that all data transfer occurs on the falling edge of the clock pulse.

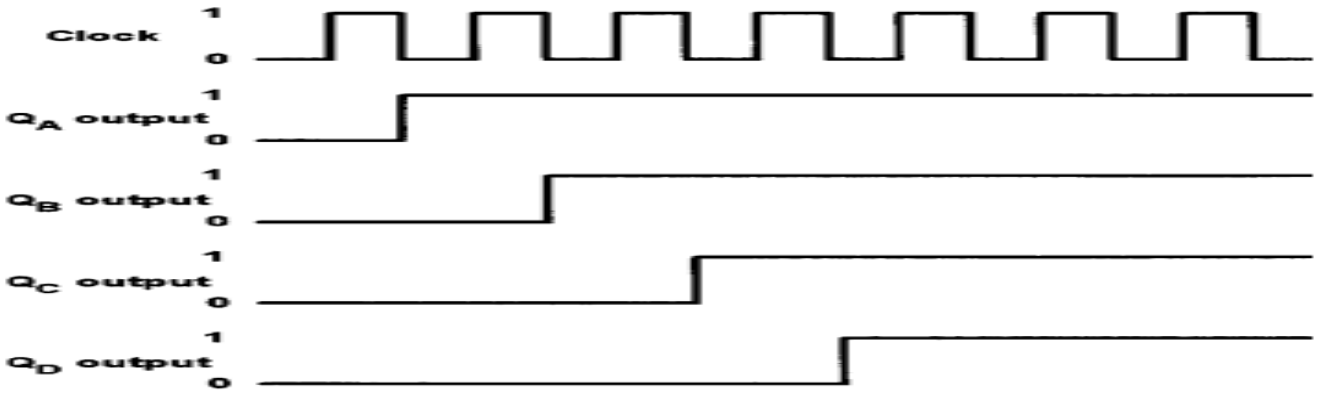


Figure 10.25 Timing diagram for the four-stage shift register shown in Fig. 10.24

Example 10.4: A logic arrangement has to be designed so that it produces the pulse train shown in Fig. 10.27. Devise a logic circuit arrangement that will generate this pulse train from a regular square wave input.

Problems

10.1 Show how a four-input AND gate can be made from three two-input AND gates.
10.2 Show how a four-input OR gate can be made from three two-input OR gates.
10.3 Construct the truth table for the logic gate arrangement shown in Fig. 10.37.
10.4 Using only two-input NAND gates, show how each of the following logical functions can be satisfied: (a) two-input AND; (b) two-input OR; (c) four-input AND. In each case, use the minimum number of gates. (Hint: a two-input NAND gate can be made into an inverter by connecting its two inputs together)
10.13 With the aid of a diagram, explain how a three-stage binary counter can be built using J-K bistables
10.14 With the aid of a diagram, explain how a three-stage shift register can be built using J-K bistables