**21CS33-DIGITAL DESIGN AND COMPUTER ORGANIZATION INTEGRATED LABORATORY**

# Course objectives:

This laboratory course enables students to get practical experience in design, assembly, and evaluation/testing of

1. Combinational logic circuits.
2. Flip - Flops and their operations
3. Counters and registers using flip-flops.
4. Synchronous sequential circuits.

# Course Outcomes

|  |  |
| --- | --- |
| CO1 | Develop the understanding of gates and circuits |
| CO2 | Minimize and Implement boolean functions |
| CO3 | Design and implement various combinational circuits |
| CO4 | Implement various sequential circuits like registers and counters |

|  |  |  |
| --- | --- | --- |
| **Exp t No** | **Content of the Experiment** | **CO** |
| 1 | 1. Realize the basic gates logic using universal gates 2. Given a Boolean function, use the KMAP technique to simplify it and verify the respective circuit designed. | CO1 & CO2 |
| 2 | Given a 4-variable logic expression, simplify the given Boolean functions and realize the simplified logic expression using 8:1 multiplexer IC 74151. Consider A, B, C as select lines and D as Data Input. | CO3 |
| 3 | 1. Implement a boolean function using an appropriate demultiplexer IC 74154. Realize the expression. 2. Design a BCD to seven segment decoder using the IC 7448 | CO3 |
| 4 | Design and implement Half adder, Full Adder, Half Subtractor, Full Subtractor using basic/universal gates. | CO3 |
| 5 | Design and implement  a) 1-bit magnitude comparator  b) Binary to Gray Code Converter using basic gates. | CO3 |

|  |  |  |
| --- | --- | --- |
| 6 | Design 4 bit Shift registers and demonstrate the working of various shift modes(SISO, SIPO, PIPO) using D flipflop IC | CO4 |
| 7 | Design a 4-bit Asynchronous Ripple Up and Down counter | CO4 |
| 8 | Design and implement a mod-n (n<8) synchronous up counter using J-K Flip-Flop ICs and demonstrate its working | CO4 |

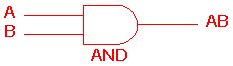
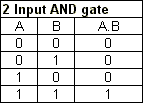
**Textbooks:**

1. Donald P Leach, Albert Paul Malvino & Goutam Saha: “**Digital Principles and Applications”**, 7th Edition, Tata McGraw Hill, 2010.
2. Carl Hamacher, Zvonko Vranesic, Safwat Zaky: “**Computer Organization”**, 5th Edition, Tata McGraw Hill, 2002.

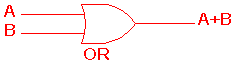
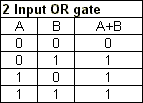
**Solution Manual:**

1. a) Realize the basic gates logic using NOR gate

**AND gate**

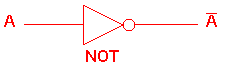
* + 1.  
    2. The AND gate is an electronic circuit that gives a **high** output (1) only if **all** its inputs are high.  A dot (.) is used to show the AND operation i.e. A.B.  Bear in mind that this dot is sometimes omitted i.e. AB

**OR gate**

* + 1. 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.

**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.

To implement basic gates using universal gates, The AND gate, OR gate and NOT gate can be implemented by using NOR gates as below:

Diagram

Description automatically generated

Diagram

Description automatically generated

1. Given a Boolean function, use the KMAP technique to simplify it and verify the respective circuit designed. (Any boolean function will be given during exam)

**Rules for k-map simplification:**

* Groups may not contain zero
* We can group 1, 2,4,8, or 2 power n cells
* Each group should be as large as possible
* Cells containing 1 must be grouped
* Groups may overlap
* Opposite grouping and corner grouping is allowed
* There should be as few groups as possible

1. Consider the following equation

F (A, B, C, D) =  (5,6,9,13,15) + d (1,7,14) => Sum of product form

Construct the truth table accordingly

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | Y |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | X |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | X |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | X |
| 1 | 1 | 1 | 1 | 1 |

Now construct the KMAP and group it by applying all the rules and fine the number of essential prime implicants

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | X | 0 | 0 |
| 0 | 1 | X | 1 |
| 0 | 1 | 1 | X |
| 0 | 1 | 0 | 0 |

00

01

11

10

00

01

Green – Prime implicant 1

Yellow – Prime implicant 2

11

10

The non-simplified expression is **F (A, B, C, D) = A̅BC̅D + A̅BCD̅+AB̅C̅D +ABC̅D+ABCD**

The simplified expression in SOP form is **F (A, B, C, D) = B C + C̅ D**

Create a combinational logic circuit for this using basic gates and simulate the same as shown below

Diagram, schematic

Description automatically generated

Verify the logic circuit against the truth table.

2. Given a 4-variable logic expression, simplify the given Boolean functions and realize the simplified logic expression using 8:1 multiplexer IC 74151. Consider A, B, C as select lines and D as Data Input.

**Multiplexer** is a combinational circuit that has maximum of 2n data inputs, ‘n’ selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines.

Since there are ‘n’ selection lines, there will be 2n possible combinations of zeros and ones. So, each combination will select only one data input. Multiplexer is also called as **Mux**.

The 8x1 Multiplexer has eight data inputs I7 to I0, three selection lines s2, s1 & s0 and one output Y.

Assume that the 4-variable Boolean Function Y = F (A, B, C, D) = ∑ (2, 3,4, 5, 7, 13,15) + d (8,9,10,11).

Construct a truth table and specify the min terms

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | Y |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | X |
| 1 | 0 | 0 | 1 | X |
| 1 | 0 | 1 | 0 | X |
| 1 | 0 | 1 | 1 | X |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

**Step 3:** Draw implementation table and create K-Maps

11

bc

a

01

10

00

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 0 | 1 |
| X | X | D | D |

0

1

Three prime implicants and the final output expression is

**Y = ̅BC + A̅BC̅ + AD**

Populate the implementation table and design the circuit as below

**4-variable Boolean function design:**

Diagram, schematic

Description automatically generated

**Observation:** Verify the entire truth table and all the values

1. a) Implement a boolean function using an appropriate demultiplexer IC 74154. Realize the expression.

De-Multiplexer is a combinational circuit that performs the reverse operation of Multiplexer. It has single input, ‘n’ selection lines and maximum of 2n outputs. The input will be connected to one of these outputs based on the values of selection lines.

Since there are ‘n’ selection lines, there will be 2n possible combinations of zeros and ones. So, each combination can select only one output.

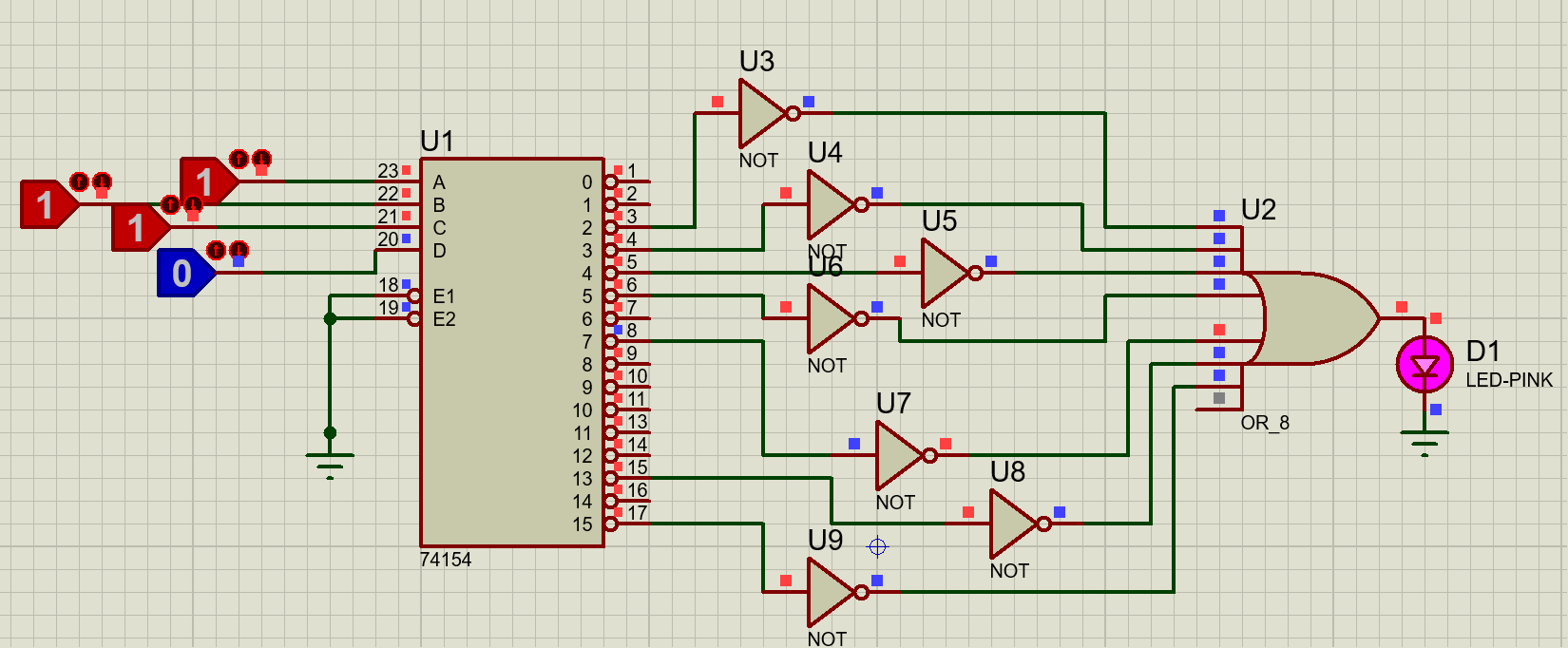
Assume a boolean function

Y = F (A, B, C, D) = ∑ (2, 3,4, 5, 7, 13,15).

Write the truth table and mark the min terms

Here E1 pin has to be connected to an input to make it work as a Demultiplexer

Draw the pin diagram as shown below and verify the truth table



b) Design a BCD to seven segment decoder using the IC 7448

The combinational circuit that change the binary information into 2N output lines is known as **Decoders.** The binary information is passed in the form of N input lines. The output lines define the 2N-bit code for the binary information. The **Decoder** performs the reverse operation of the **Encoder**. At a time, only one input line is activated for simplicity. The produced 2N-bit output code is equivalent to the binary information.



The Decoder IC 7448 accepts a 1-2-4-8 positive-logic Binary Coded Decimal (BCD) input and converts it to the proper pattern necessary to illuminate a 7 segment display. A high output is intended to light the segment. (Common cathode), meaning the negative connection of all of the LEDs is tied together.

Design the circuit as shown below and very the BCD inputs decoded to respective decimal form

Diagram, schematic

Description automatically generated

**Data Table to be verified:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BCD Input Data** | | | | **Numerical Displayed** |
| **SW3** | **SW2** | **SW1** | **SW0** |  |
| **0** | **0** | **0** | **0** |  |
| **0** | **0** | **0** | **1** |  |
| **0** | **0** | **1** | **0** |  |
| **0** | **0** | **1** | **1** |  |
| **0** | **1** | **0** | **0** |  |
| **0** | **1** | **0** | **1** |  |
| **0** | **1** | **1** | **0** |  |
| **0** | **1** | **1** | **1** |  |
| **1** | **0** | **0** | **0** |  |
| **1** | **0** | **0** | **1** |  |

1. Design and implement Half adder, Full Adder, Half Subtractor, Full Subtractor using basic/universal gates.

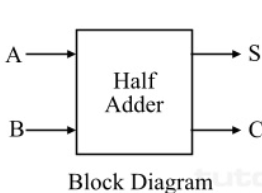
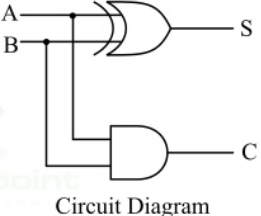
**Adders** and **subtractors** both are the combinational logic circuits that can add or subtract numbers, more specifically binary numbers.

Adders are classified into two types namely −

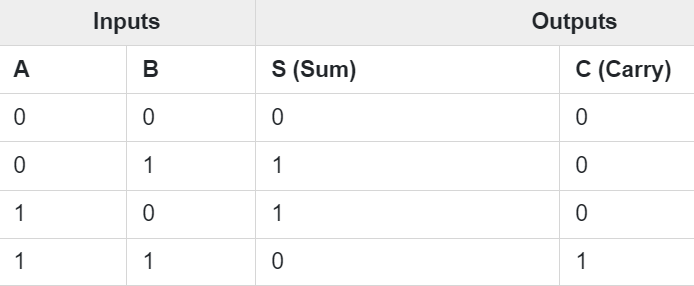
* Half Adder
* Full Adder

A **half-adder** is a combinational logic circuit that performs the addition of only two bits (binary digits). Whereas, a **full-adder** is a combination circuit that performs three bits (binary digits), where two are the significant bits and one is a carry from previous execution.

The half adder circuit is designed by connecting an EX-OR gate and one AND gate. It has two input terminals and two output terminals for sum and carry.

The truth table of half adder is



The characteristic equations of half adder, i.e., equations of sum (S) and carry (C) are obtained according to the rules of binary addition. These equations are given below −

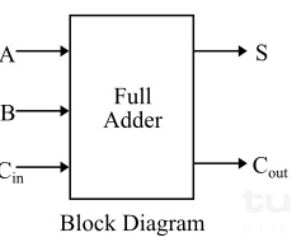
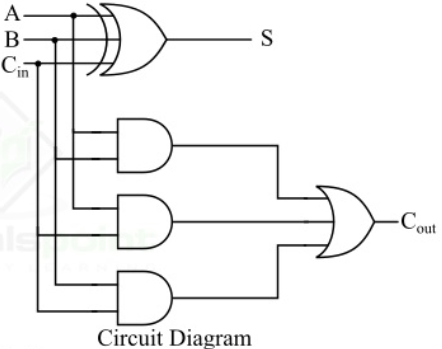
The sum (S) of the half-adder is the XOR of A and B. Thus,

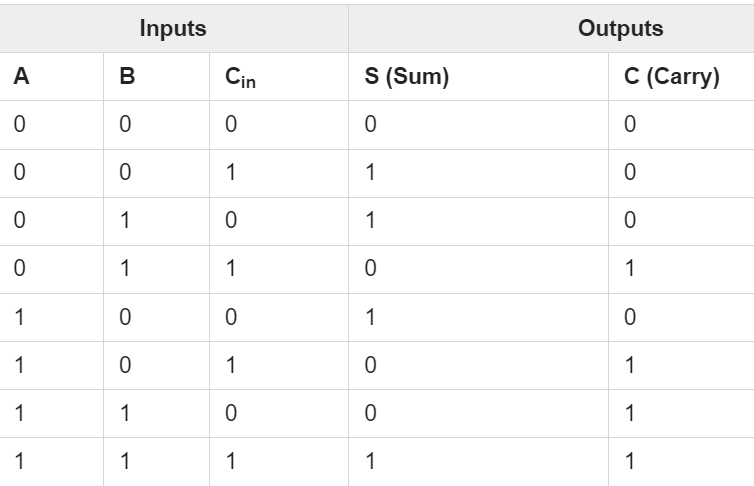
Sum, S=A⊕B=AB′+A′B

The carry (C) of the half-adder is the AND of A and B. Therefore,

Carry, C=A⋅B

A combinational logic circuit that can add two binary digits (bits) and a carry bit, and produces a sum bit and a carry bit as output is known as a **full-adder**.



**Characteristic Equations of Full Adder**

The characteristic equations of the full adder, i.e. equations of sum (S) and carry output (Cout) are obtained according to the rules of binary addition. These equations are given below −

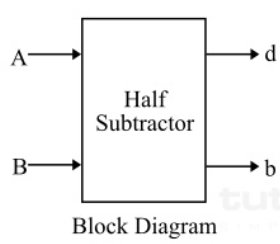
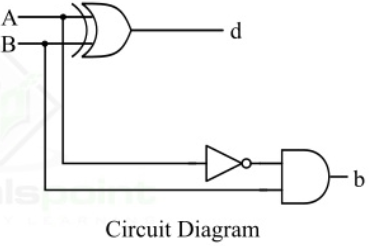
The sum (S) of the full-adder is the XOR of A, B, and Cin. Therefore,

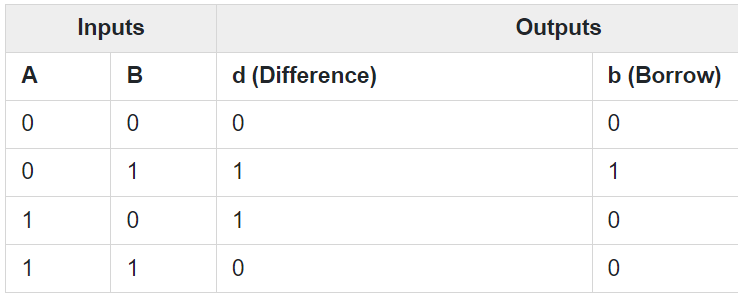
Sum, S=A⊕B⊕Cin = A′B′Cin+A′BC′in+AB′C′in+ABCin

The carry (C) of the half-adder is the AND of A and B. Therefore,

Carry, Cout = AB+ACin+BCin

A **half-subtractor** is a combinational logic circuit that have two inputs and two outputs (i.e. difference and borrow). The half subtractor produces the difference between the two binary bits at the input and also produces a borrow output (if any). In the subtraction (A-B), A is called as **Minuend bit** and B is called as Subtrahend bit.



**Characteristic Equation of Half Subtractor**

The characteristic equations of the half subtractor, are obtained by following the rules of binary subtraction. These equations are given below −

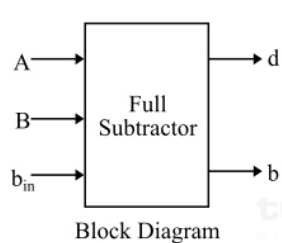
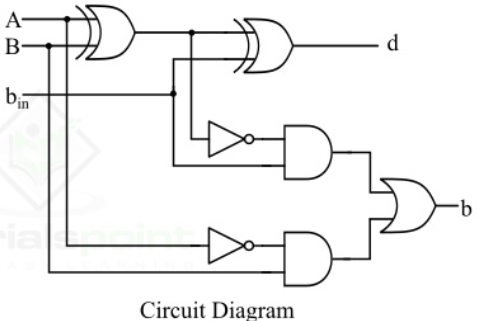
The difference (d) of the half subtractor is the XOR of A and B. Therefore,

**Difference, d=A⊕B=A′B+AB′**

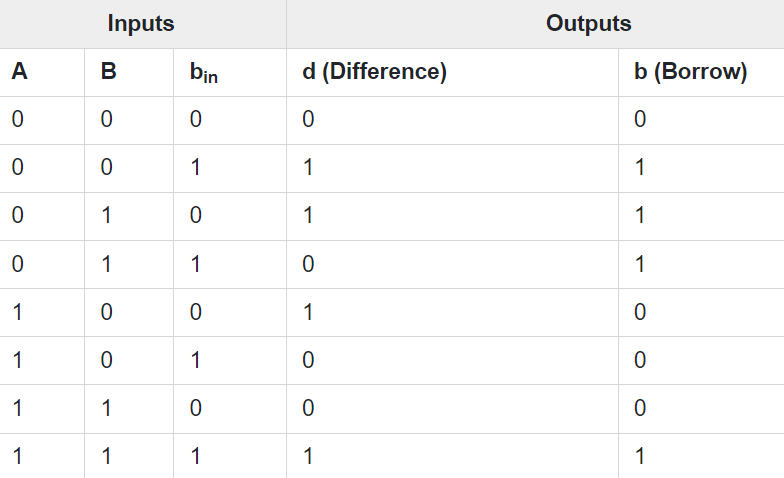
The borrow (b) of the half subtractor is the AND of A’ (compliment of A) and B. Therefore,

**Borrow, b=A′B**

A **full-subtractor** is a combinational circuit that has three inputs A, B, bin and two outputs d and b. Where, A is the minuend, B is subtrahend, bin is borrow produced by the previous stage, d is the difference output and b is the borrow output.

****  ****

### **Truth Table of Full-Subtractor**

****

**Characteristic Equations of Full Subtractor**

The characteristic equations of the full subtractor

The difference (d) of the full subtractor is the XOR of A, B, and bin. Therefore,

Difference, d = A⊕B⊕bin=A′B′bin+AB′b′in+A′Bb′in+ABbin

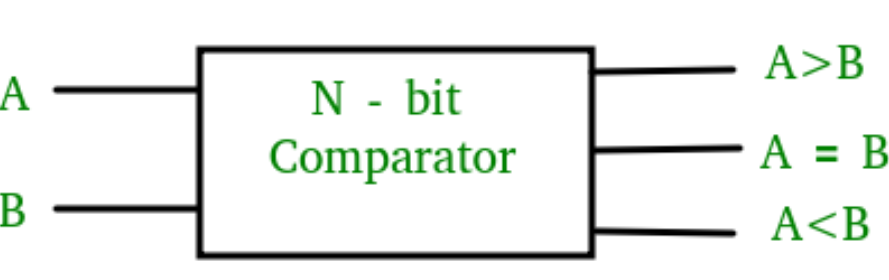
The borrow (b) of the full subtractor is given by,

Borrow, b = A′B′bin+A′Bb′in+A′Bbin+ABbin

5) Design and implement

a) 1-bit magnitude comparator

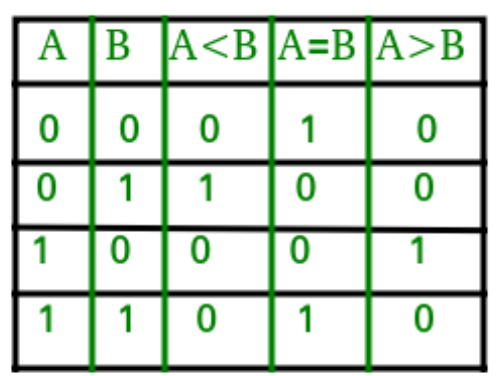
A magnitude digital Comparator is a combinational circuit that **compares two digital or binary numbers** in order to find out whether one binary number is equal, less than, or greater than the other binary number. It has two inputs one for A and the other for B and have three output terminals, one for A > B condition, one for A = B condition, and one for A < B condition.



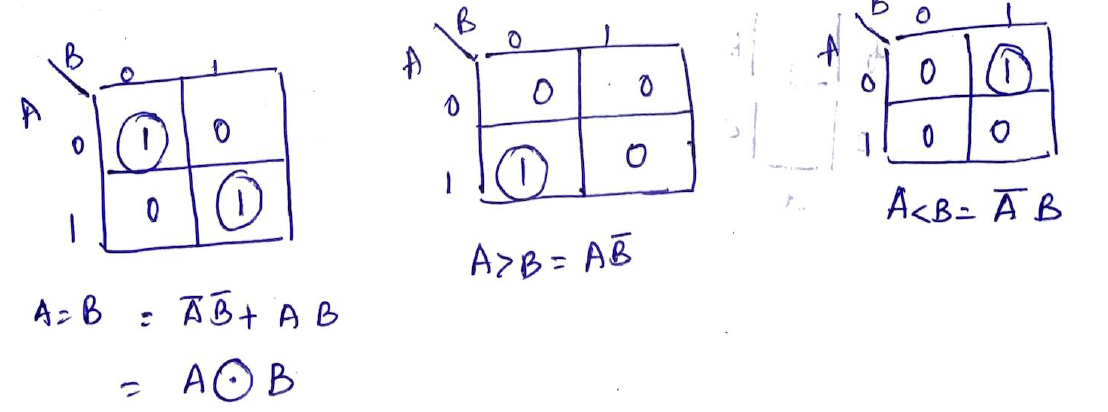
### **1-Bit Magnitude Comparator:**

A comparator used to compare two bits is called a single-bit comparator. It consists of two inputs each for two single-bit numbers and three outputs to generate less than, equal to, and greater than between two binary numbers.

The truth table for a 1-bit comparator is given below:



Draw KMaps and derive the expression for these

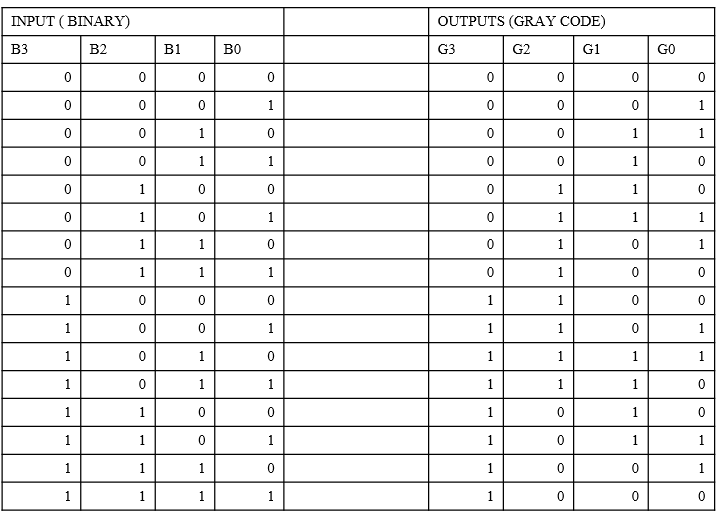


Design the circuit accordingly and verify the truth table

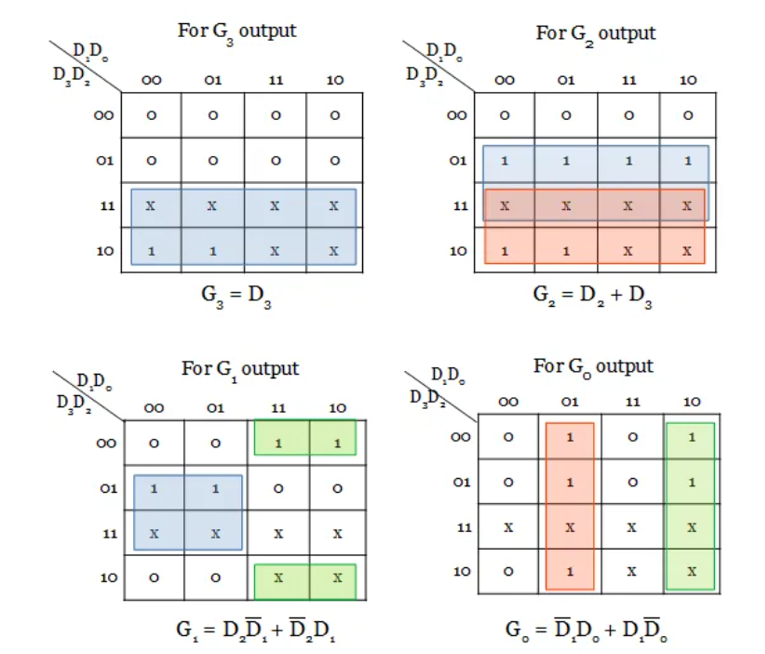
b) Binary to Gray Code Converter using basic gates

* A code converter is **a logic circuit that changes data presented in one type of binary code to another type of binary code**, such as BCD to binary, BCD to 7−segment, binary to BCD, BCD to XS3, binary to Gray code, and Gray code to binary.
* The reflected binary code (RBC), also known as reflected binary (RB) or Gray code after Frank Gray, is an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit).
* For example, the representation of the decimal value "1" in binary would normally be "001" and "2" would be "010". In Gray code, these values are represented as "001" and "011". That way, incrementing a value from 1 to 2 requires only one bit to change, instead of two.
* Gray codes are widely used to prevent spurious output from electromechanical switches and to facilitate error correction in digital communications such as digital terrestrial television and some cable TV systems.

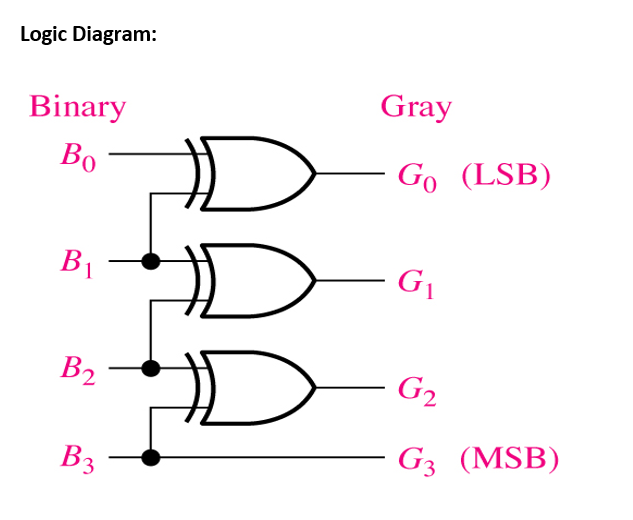
Truth table for Binary to Gray code conversion is



Draw the KMaps to find the logic expressions

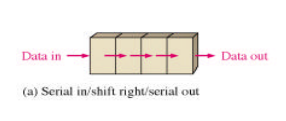
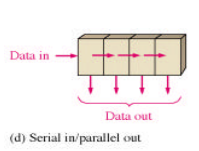
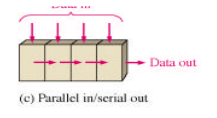
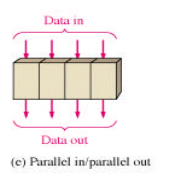


Design the Logic Circuit and verify the truth table



6.Design 4-bit Shift registers and demonstrate the working of various shift modes (SISO, SIPO, PIPO) using D flipflop IC

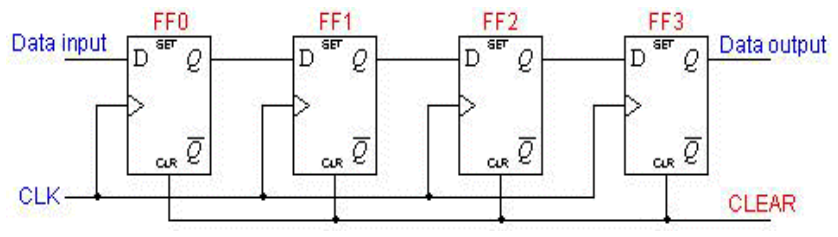
Shift registers are a type of sequential logic circuit, mainly for storage of digital data. They are a group of flip-flops connected in a chain so that the output from one flip-flop becomes the input of the next flip-flop. Most of the registers possess no characteristic internal sequence of states. All flip-flop is driven by a common clock, and all are set or reset simultaneously. In these few lectures, the basic types of shift registers are studied, such as Serial In - Serial Out, Serial In - Parallel Out, Parallel In – Serial Out, Parallel In - Parallel Out

**Serial In - Serial Out Shift Registers**

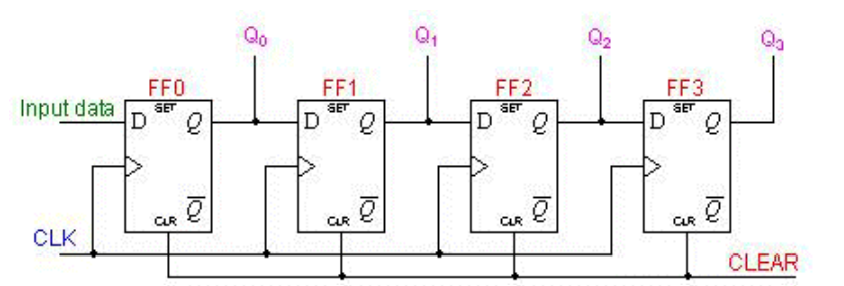
The serial in/serial out shift register accepts data serially – that is, one bit at a time on a single line. It produces the stored information on its output also in serial form.

four-bit shift register



**Serial In - Parallel Out Shift Registers**

For this kind of register, data bits are entered serially in the same manner as discussed in the last section. The difference is the way in which the data bits are taken out of the register. Once the data are stored, each bit appears on its respective output line, and all bits are available simultaneously. A construction of a four-bit serial in - parallel out register is shown below.



**Parallel In – Serial Out Shift Registers**

* + - Data bits are entered in parallel fashion.
    - The circuit shown below is a four-bit parallel input serial output register.
    - Output of previous Flip Flop is connected to the input of the next one via a combinational circuit.
    - The binary input word B0, B1, B2, B3 is applied though the same combinational circuit.

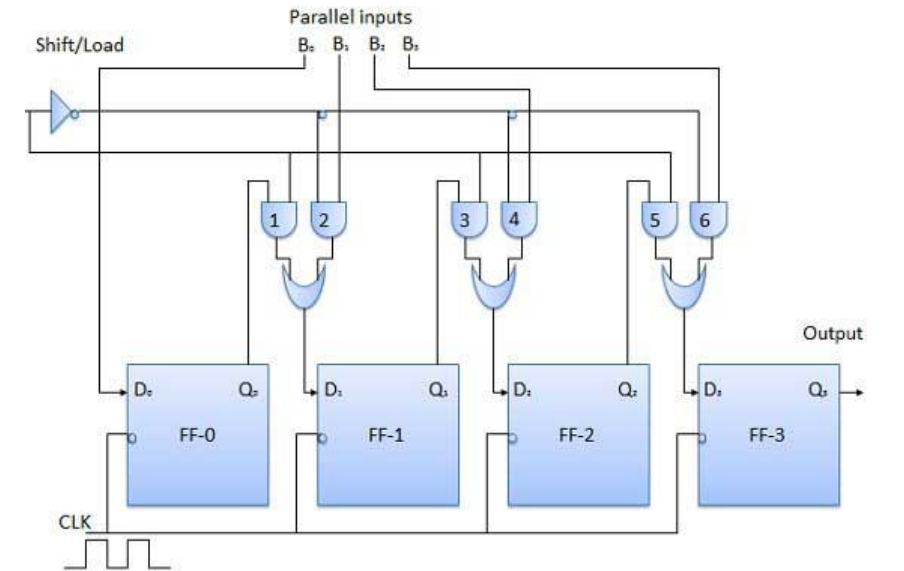
There are two modes in which this circuit can work namely - shift mode or load mode.

**Load mode**

When the shift/load bar line is low (0), the AND gate 2, 4 and 6 become active they will pass B1, B2, B3 bits to the corresponding flip-flops. On the low going edge of clock, the binary input B0, B1, B2, B3 will get loaded into the corresponding flip-flops. Thus parallel loading takes place.

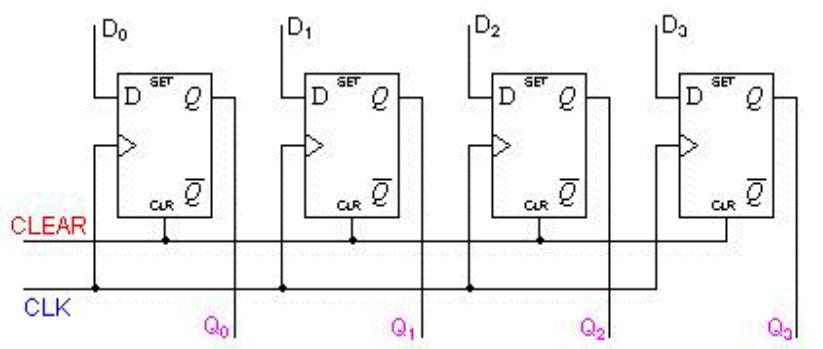
**Shift mode**

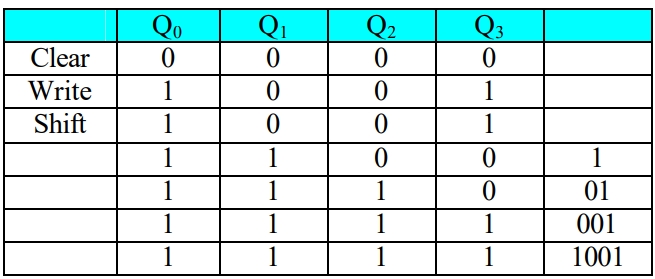
When the shift/load bar line is low (1), the AND gate 2, 4 and 6 become inactive. Hence the parallel loading of the data becomes impossible. But the AND gate 1,3 and 5 become active. Therefore, the shifting of data from left to right bit by bit on application of clock pulses. Thus the parallel in serial out operation takes place.

****

**Parallel In - Parallel Out Shift Registers**

For parallel in - parallel out shift registers, all data bits appear on the parallel outputs immediately following the simultaneous entry of the data bits. The following circuit is a four-bit parallel in - parallel out shift register constructed by D flip-flops.





7. Design a 4-bit Asynchronous Ripple Up and Down counter

Ripple counter is a special type of **Asynchronous** counter in which the clock pulse ripples through the circuit. The n-MOD ripple counter forms by combining n number of flip-flops. The n-MOD ripple counter can count 2n states, and then the counter resets to its initial value.

**Features of the Ripple Counter:**

* It is an example of an asynchronous counter.
* The flip flops are used in toggle mode.
* The external clock pulse is applied to only one flip flop. The output of this flip flop is treated as a clock pulse for the next flip flop.
* In counting sequence, the flip flop in which external clock pulse is passed, act as LSB.

**Up Counter**

The up-counter counts the states in ascending order.

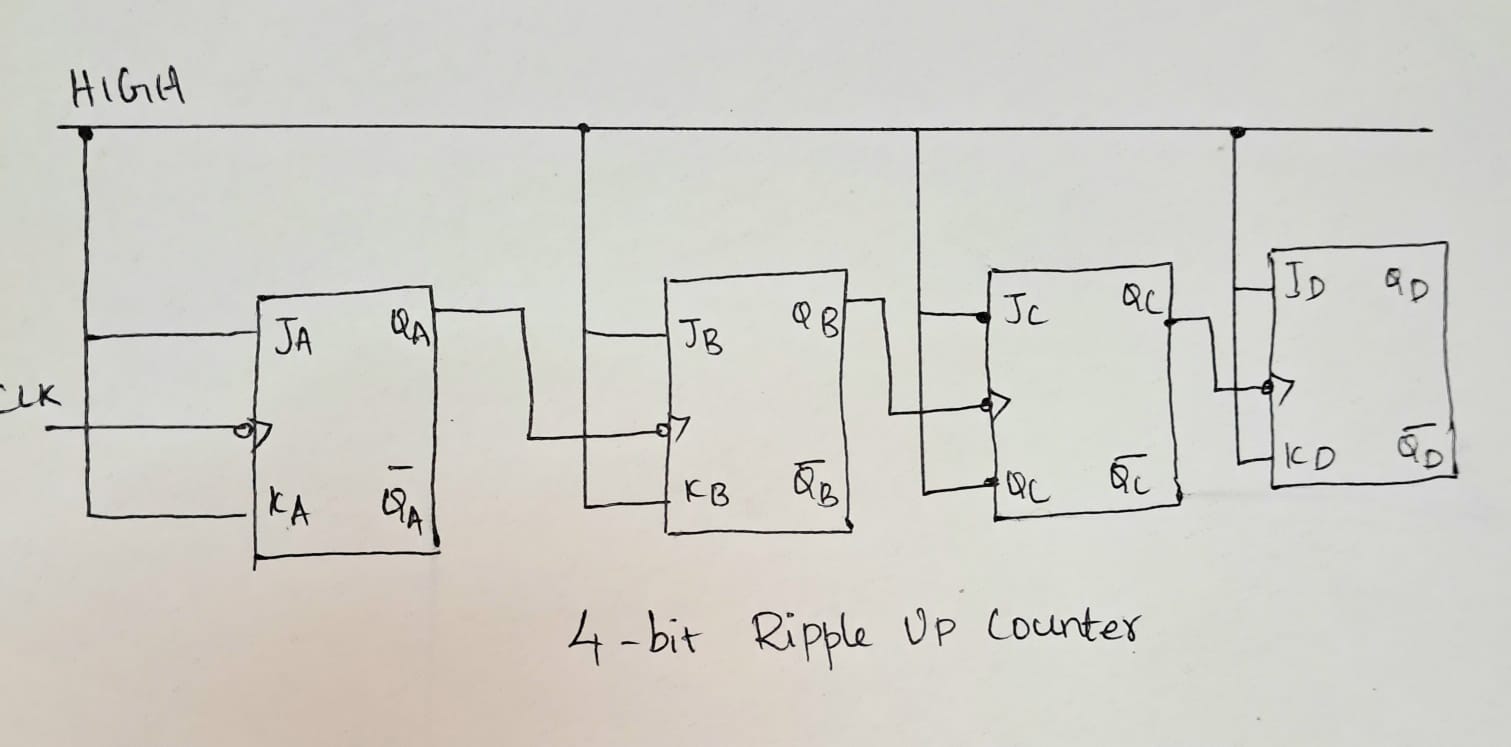
**Down Counter**

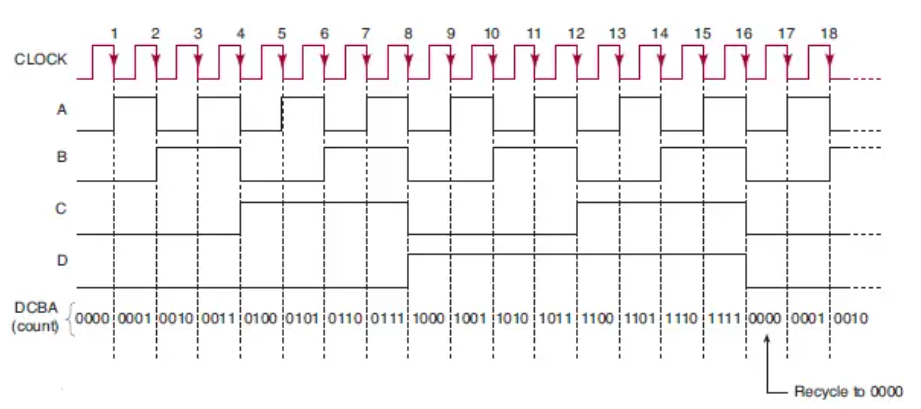
The down counter counts the states in descending order.

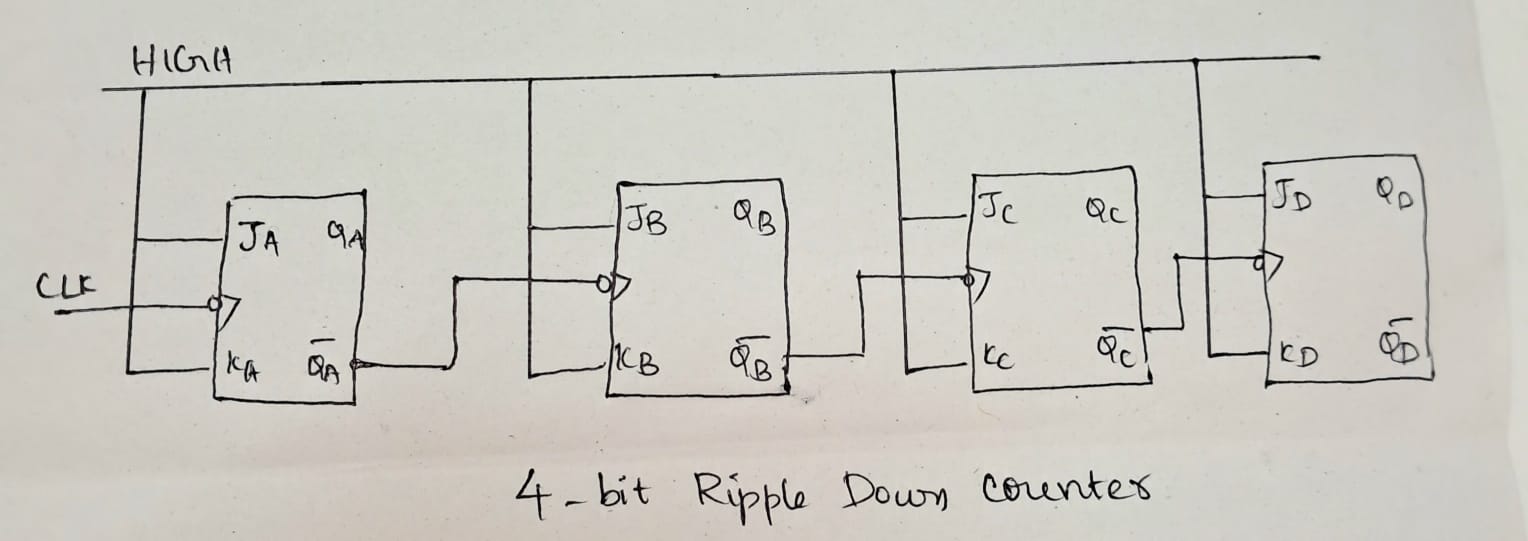
**Up-Down Counter**

The up and down counter is a special type of bi-directional counter which counts the states either in the forward direction or reverse direction. It also refers to a reversible counter.

A 4-bit Ripple Up counter takes the values from 0000 and goes till 1111 and the down counter counts in reverse direction







Similarly draw the waveforms for the down counter also

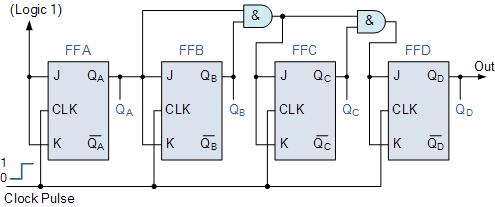
8.Design and implement a mod-n (n<8) synchronous up counter using J-K Flip-Flop ICs and demonstrate its working

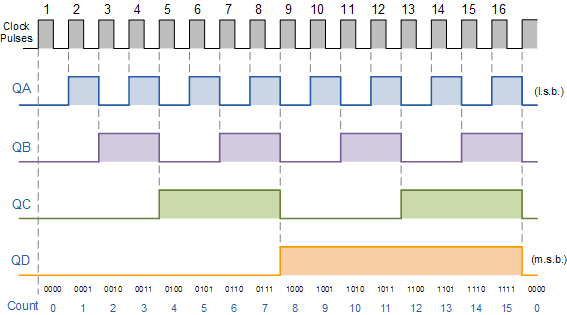
In the asynchronous counters the output of one stage is connected directly to the clock input of the next counter stage in the chain. But in the synchronous counter all its stages are clocked together at the same time.

The problem with Asynchronous counters is that they suffer from what is known as “Propagation Delay” in which the timing signal is delayed a fraction through each flip-flop.

However, with the **Synchronous Counter**, the external clock signal is connected to the clock input of every individual flip-flop within the counter so that all of the flip-flops are clocked together simultaneously (in parallel) at the same time

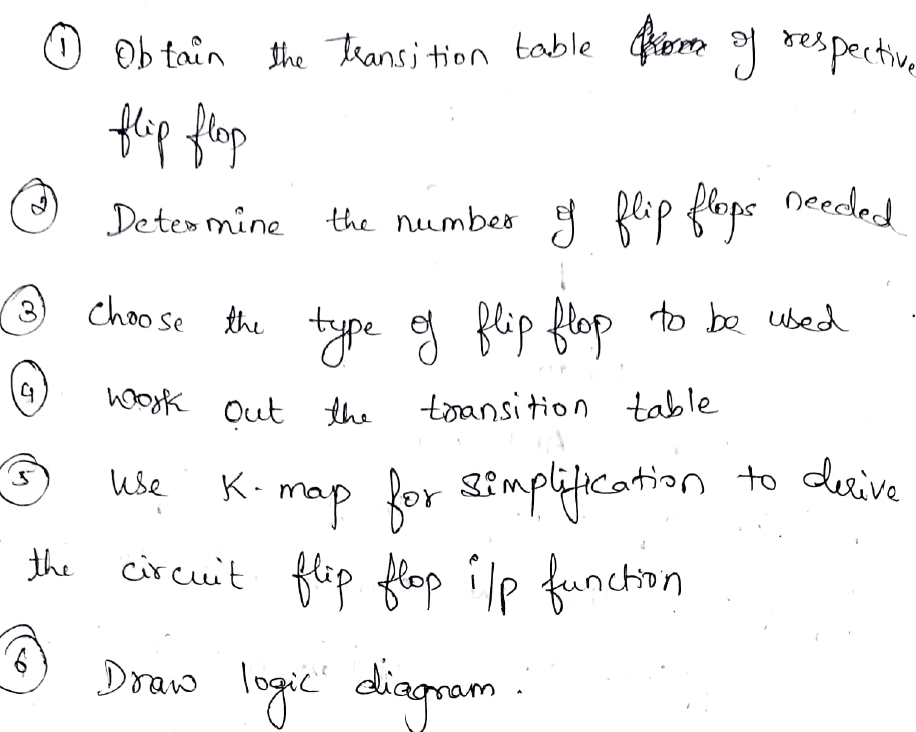
### Binary 4-bit Synchronous Up Counter (Any mod counter can be asked in the exam)





Because this 4-bit synchronous counter counts sequentially on every clock pulse the resulting outputs count upwards from 0 (0000) to 15 (1111). Therefore, this type of counter is also known as a **4-bit Synchronous Up Counter**

For any mod counter asked in the exam, follow the steps below and design the counter accordingly. Refer notes shared for examples.



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*