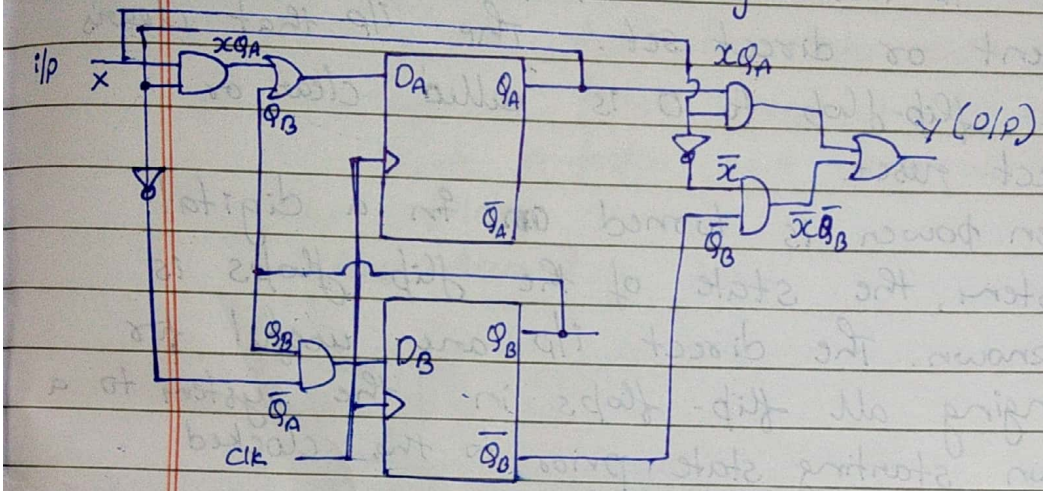
Akshatha. Y. C
4AL18ECO05

SURYA Gold
Date Page

• DAY 2: LOGIC DESIGN:

Analysis of Clocked Sequential Circuits (with D-FF)

To find State diagram:



STEP 1: Find out input and o/p equations.

$D_A = xQ_A + Q_B$

$D_B = Q_B \cdot \bar{Q}_A$

$Y = \bar{x}\bar{Q}_B + xQ_A$

$D = Q_{n+1}$

$Q_A^+ = D_A$

$Q_B^+ = D_B$

Step 2: State table.

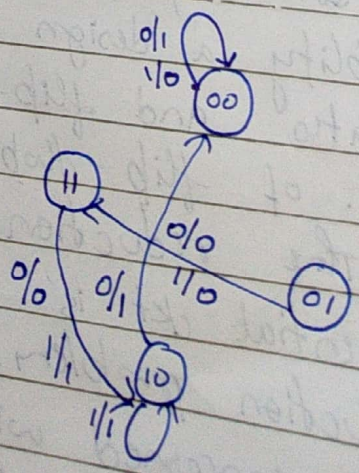
P.S			N.S		
Q_A	Q_B	x	Q_A^+	Q_B^+	y
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	1	0
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	ϕ	0	1
1	1	0	1	0	0
1	1	1	1	0	1

Step 3: State diagram:

There are 4 states:

$S_0 = 00$ $S_2 = 10$

$S_1 = 01$ $S_3 = 11$



→ Some flip-flops have asynchronous i/p that are used to force the flip-flop to a particular state independently of the clock.

→ The i/p that sets the flip-flop to 1 called preset or direct set. The i/p that clears the flip-flop to 0 is called clear or direct reset.

→ When power is turned on in a digital system, the state of the flip-flops is unknown. The direct i/p are useful for bringing all flip-flops in the system to a known starting state prior to the clocked operation.

- Positive ~~Edge~~ Edge triggered D-FF
  - A ckt diagram of a positive edge triggered D F-F is shown below. It has an additional reset i/p connected to the 3 NAND gates.

- State Reduction And Assignment.
  - Two sequential ckts may exhibit the same i/p - o/p behaviour but have a different number of internal states in their state diagram.
  - Certain properties of sequential ckt may simplify a design by reducing the no. of gates and flip-flop it uses. Reducing the no. of flip-flop reduces the cost of a ckt.
  - The reduction in the no. of FF in a sequential ckt is referred to as the state reduction problem. State reduction algorithms are concerned with procedures for reducing the number of states in a stable table.

- **Python:**

→ Object Oriented programming Explained.
It is just to over-rise the code.

→ Turning the Application into OOP style, Part-1

→ Turning the Application into OOP style, Part-2.

→ Creating a bank account object.
Create a folder and put all the files in that folder.

```
1. class- Account:
2.
3. def_ init_ (self, filepath) then balance .txt.
                                    1000
grab this value.
4. with open (filepath, 'r') as file:
self.balance= int(file.read())

7. account = Account("account / balance.txt")
8. print (account).
```

o/p: Account object at 0x00580750>

→ Inheritance: The process of creating a new class in a base class.
we have transfer the amount we have to add transfer method.

```
Class checking (Account):
def_ init_ (self, file path, fee):
  Account_ init_ (self, file path)
  self, fee=fee.
```

→ OOP Glossary:
GUI in OOP Design.
After the script containing the GUI code by functional - Oriented design into an oop design For you convenience, the field fronted.py.

Soln:

```
#fronted.py
from thinker import*
from backend import database
database = Database("books.db")
Class window (object):
    def - init - (self, window):
    self. window = window)
    self. window. wn =title ("Bookstore")
    l1 = Label (window, text = "title")
   l2 = gird (row=0, column=0)
    12 = label (window, text = "Author").
    13 = label (window, text = "year")
    14 = label (window, text = "ISBN")
    self. author - text = stringvar ()
    self . year_text - string ()
window = Tk()
window (window)
window. main loop().
```