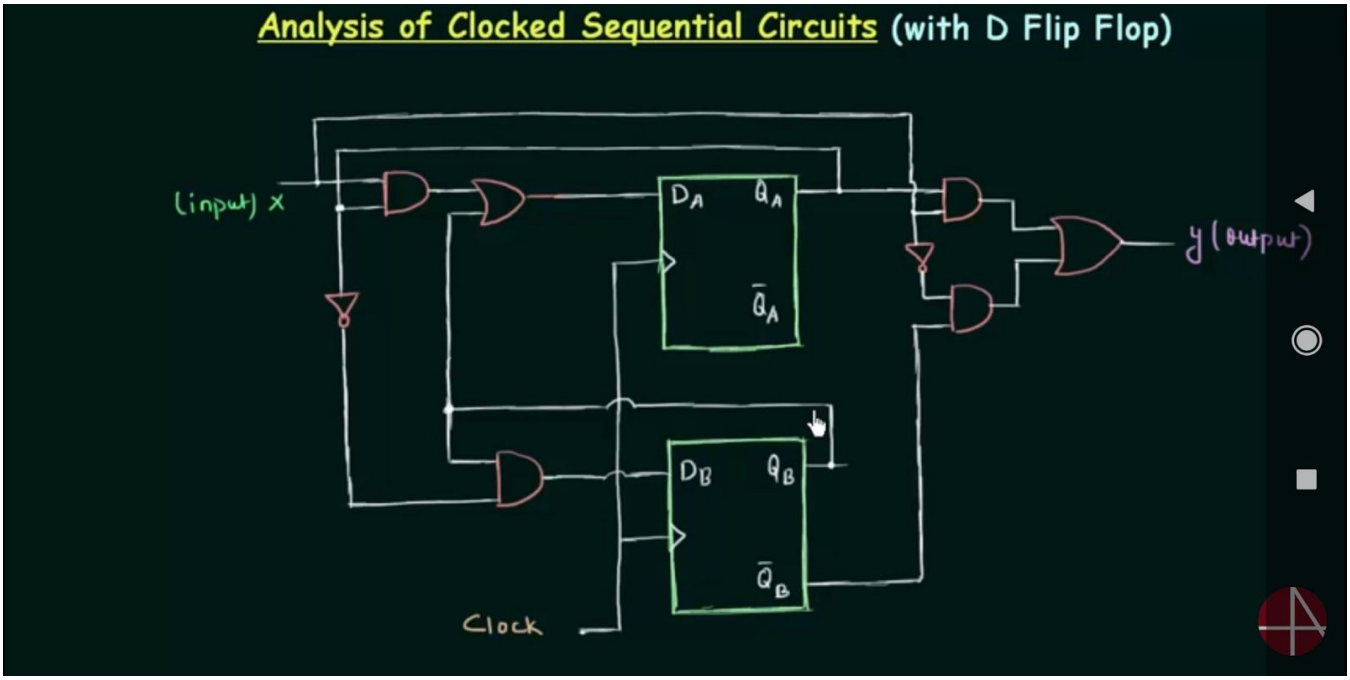
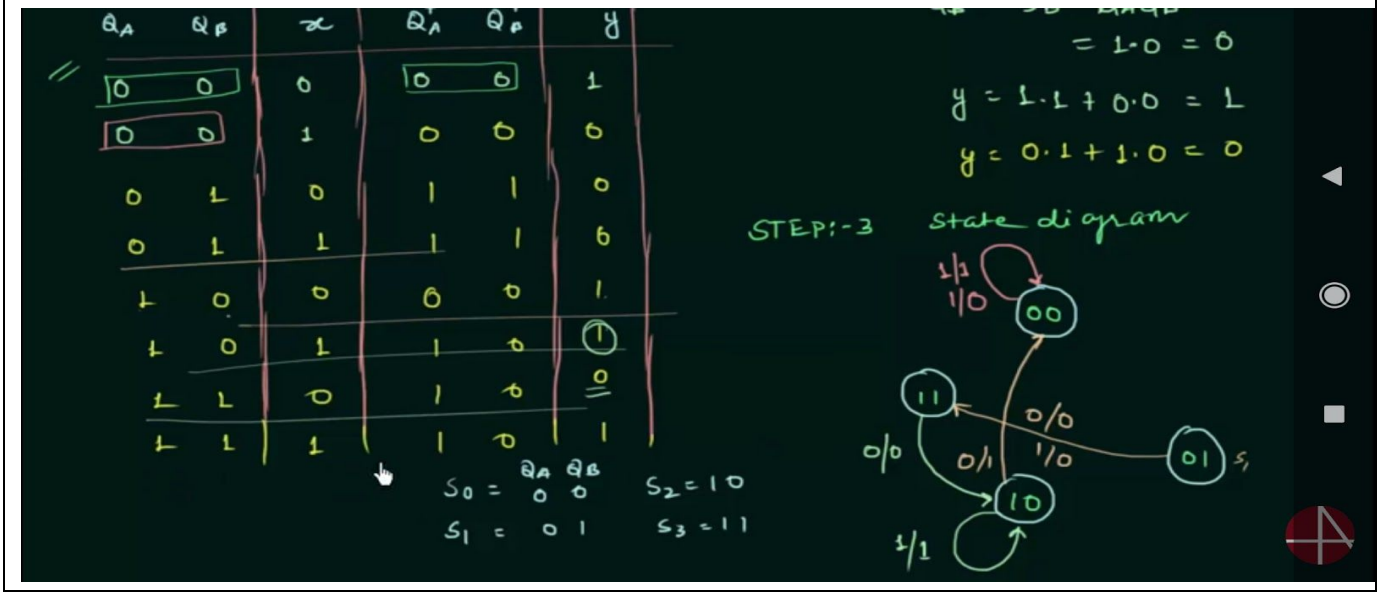
DAILY ASSESSMENT FORMAT

Date:	**29-05-2020**	**Name:**	**MOUNITHA D M**
Course:	**LOGIC DESIGN**	**USN:**	**4AL17EC055**
Topic:	**Analysis of Clocked Sequential Circuits Digital Clock Design**	**Semester & Section:**	**6TH SEM "A" SEC**
Github Repository:	**Mounitha_-ec055**		

FORENOON SESSION DETAILS

Image of session





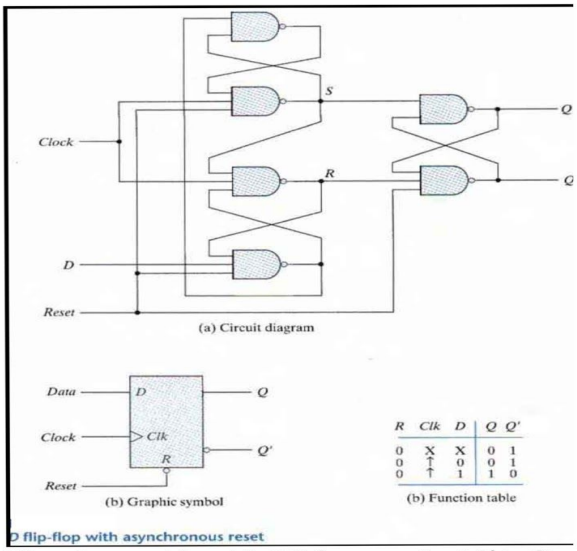
upscfever.com/upsc-fever/er

ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS

- Some flip-flops have asynchronous inputs that are used to force the flip-flop to a particular state independently of the clock
- The input that sets the **flip-flop to 1 is called preset or direct set.** The input 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 inputs are useful for bringing all flip-flops in the system to a known starting state prior to the clocked operation.
- The knowledge of the type of flip-flops and a list of the Boolean expressions of the combinational circuit provide the information needed to draw the logic diagram of the sequential circuit. The part of the combinational circuit that gene rates external outputs is described algebraically by a set of Boolean functions called **output equations.** The part of the circuit that generates the inputs to flip-

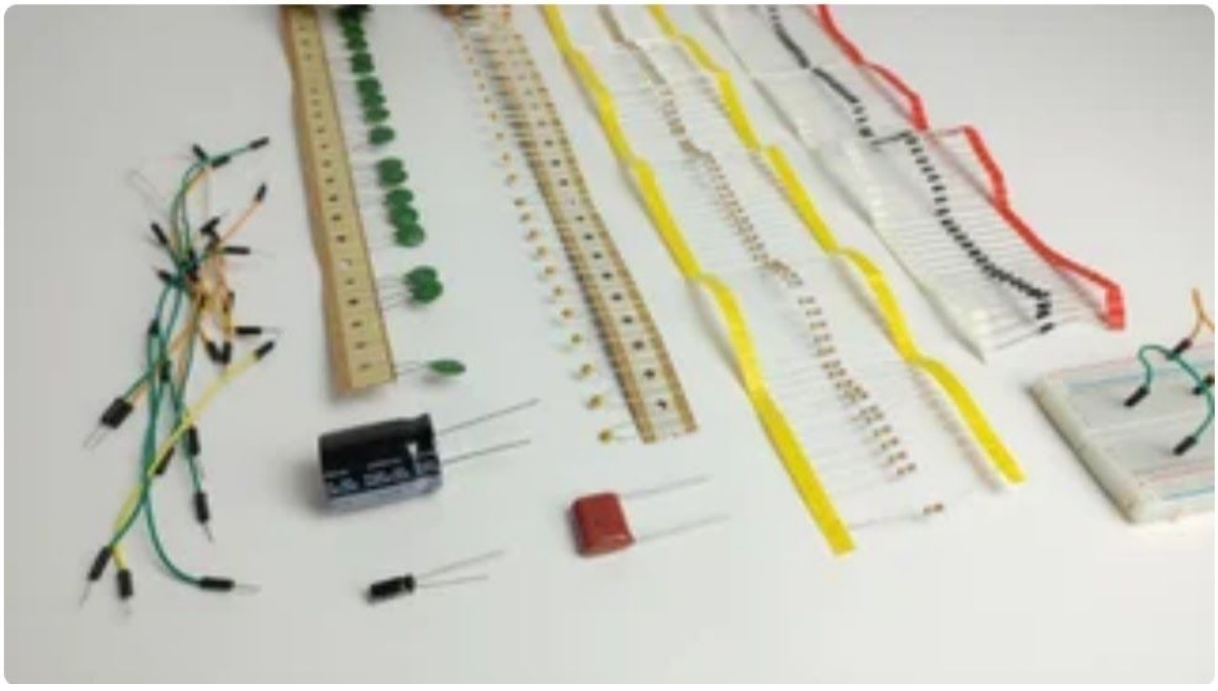
ositive Edge Triggered D Flip-flop

- A circuit diagram of a Positive edge triggered D Flip-flop is shown as below. It has an **additional reset input** connected to the three NAND gates.



D flip-flop with asynchronous reset

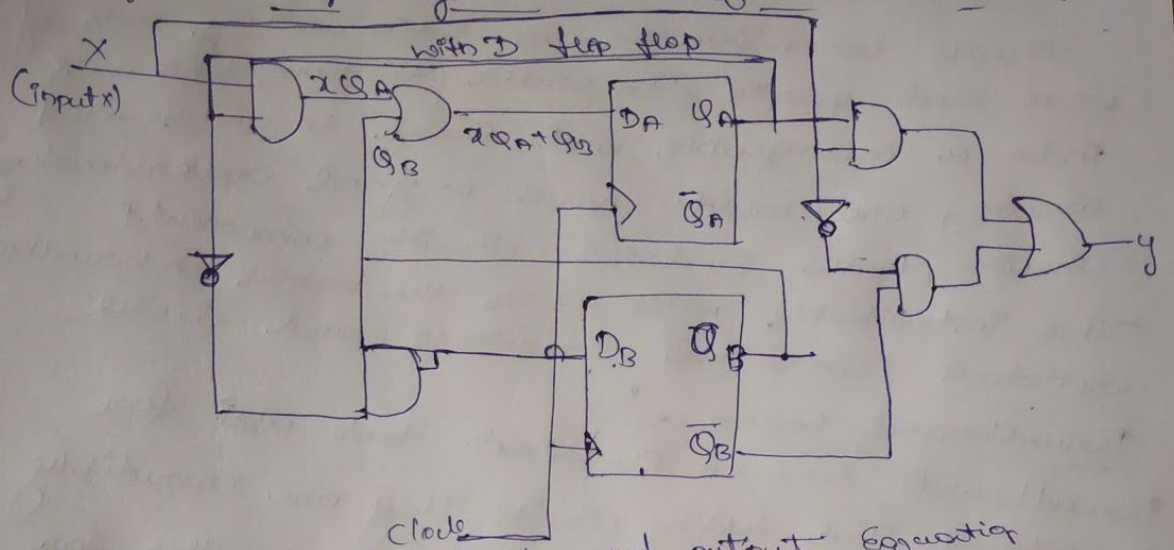
- When the **reset input is 0 it forces output Q' to Stay at 1** which clears output Q to 0 thus resetting the flip-flop.



Report – Report can be typed or hand written for up to two pages.

Logic Design 29/03/2020

Day 2 Analysis of clocked Sequential circuits



Step 1 Find out the input and output Equation

$D_A = x Q_A + Q_B$

$D_B = \bar{Q}_A Q_B$

$y = x \bar{Q}_B + x Q_A$

$D = Q_{n+1}$

$Q_A^+ = D_A$

$Q_B^+ = D_B$

Step 2 State table

Present State | Next State

Q_A	Q_B	x	Q_A^+	Q_B^+	y
0	0	0	0	0	1
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	1	0	1
1	1	0	1	0	1
1	1	1	1	0	0
				0	1

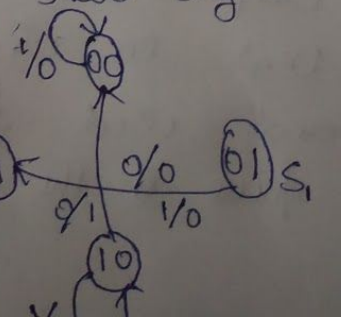
$Q_A^+ = D_A = x Q_A + Q_B$
$= 0.0 + 0 = 0$

$Q_B^+ = D_B = \bar{Q}_A Q_B$
$= 1.0 = 0$

$y = 1.1 + 0.0 = 1$

$y = 0.1 + 1.0 = 0$

State Digram



$S_0 = Q_A\ Q_B$ $S_2 = 10$

$S_3 = 11$

Digital clock design

Instructable

Circuits can often be an intimidating mess of wires and parts. In order to turn this mess into a manageable and easy to understand circuit, one must have a good understanding of basic Electrical circuit components. This instructable will cover the basics on breadboards, resistors, capacitors, inductors, and diodes.

Breadboard basics

Breadboards are an Electrical tool used for prototyping and testing circuits. They are incredibly useful when connecting components together for the first time and on a low budget.

A breadboard is made up of rows and columns of Electrical connections.

Resistors

Resistors are an integral part of Electrical Engineering as they provide a specified amount of Electrical resistance. They can reduce the flow of electrons.

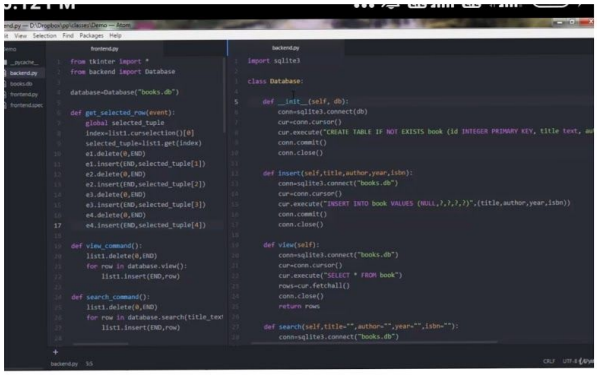
Capacitors

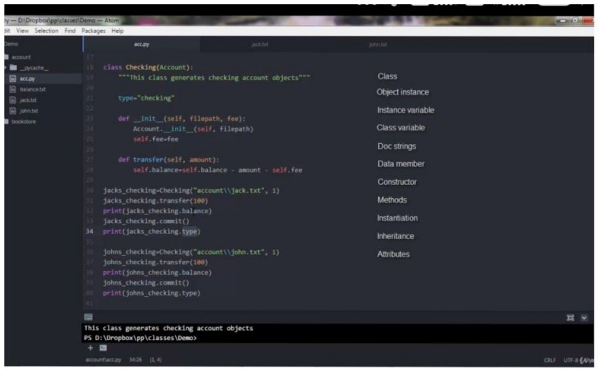
capacitors are another very useful component used in circuits. While they can be complicated and used for many different purpose we will focus on the main uses and applications.

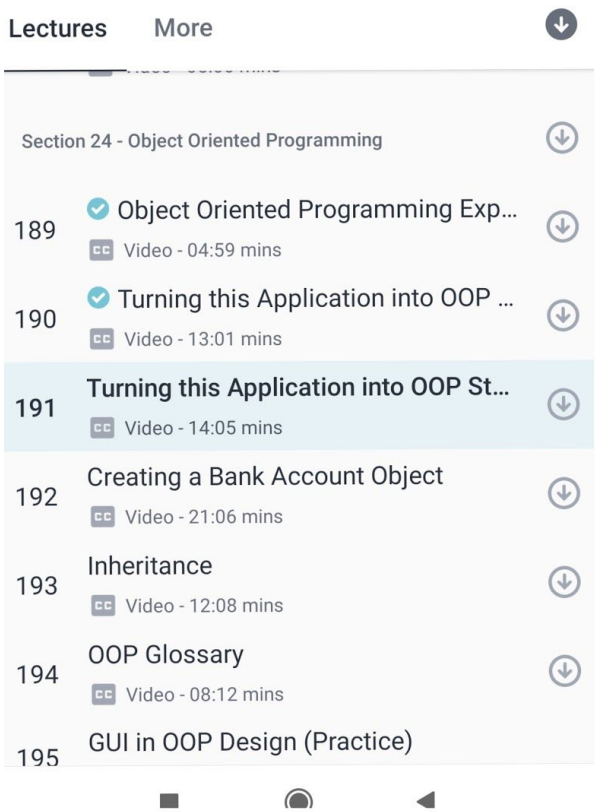
DATE	29-05-2020	Name:	MOUNITHA DM
Course:	PYTHON	USN:	4AL17EC055
Topic:	Object Oriented Programming	Semester & Section:	6TH SEM "A" SEC

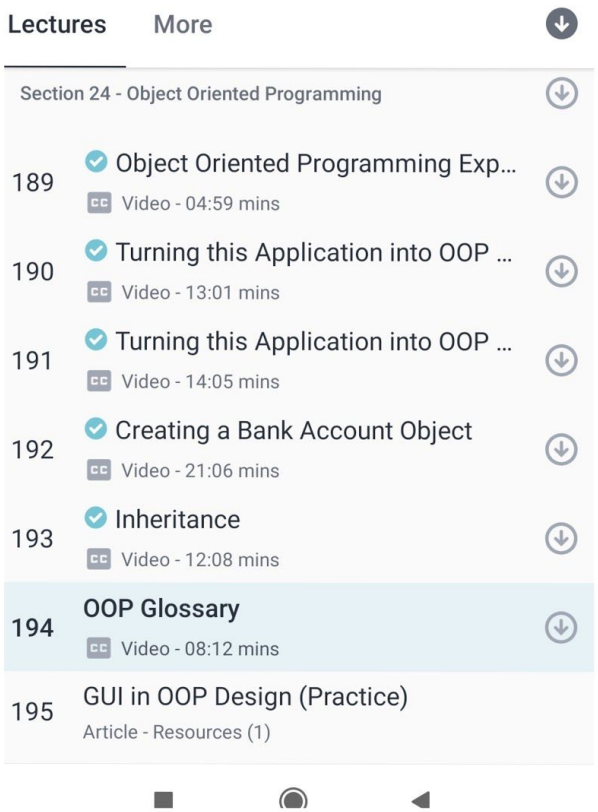
AFTERNOON SESSION DETAILS

Image of session









Report – Report can be typed or hand written for up to two pages.

Day 10 - python 29/05/2020

Section 24 - Object oriented programming.

Turning this Application into OOP Style

.frontend

```
from tkinter import *
from backend import Database
database = Database("books.db")
def get_selected_row(event):
    global selected_tuple
    index = list1.curselection()[0]
    selected_tuple = list1.get(index)
    e1.delete(0,END)
    e1.insert(END, selected_tuple[1])
    e2.delete(0,END)
    e2.insert(END, selected_tuple[2])
import sqlite3
class Database:
    def __init__(self, db):
        conn = sqlite3.connect(db)
        cur = conn.cursor()
        conn.commit()
        conn.close()
```

backend

```
class Database
    def __init__(self, db):
        conn = sqlite3.connect(db)
        self.cur = conn.cursor()
        self.cur.execute("CREATE TABLE IF NOT EXISTS book
                (id INTEGER PRIMARY KEY, title text, author
        conn.commit()
    def insert(self, title, author, year, isbn)
        self.cur.execute("INSERT INTO book VALUES(NULL,?,?,?,?)
```

Creating a Bank Account object

```
Class Account:
def __init__(self, filepath):
with open(filepath, 'r') as file:
  self.balance = int(file.read())

def withdraw(self, amount):
self.balance = self.balance - amount

def deposit(self, amount):
self.balance = self.balance + amount
account = Account("account/balance.txt")
print(account.balance)
account.withdraw(100)
print(account.balance)
```

Inheritance

```
account = Account("account/balance.txt")
print(account.balance)
account.deposite(200)
print(account.balance)
account.commit()

class checking(Account):
  def __init__(self, filepath):
  Account.__init__(self, filepath)
  def transfer(self, amount):
  self.balance = self.balance - amount
checking = checking("account/balance.txt")
checking.transfer(110)
print(checking.balance)
```

OOP Glossary

```
def transfer(self, amount):
self.balance = self.balance - amount - self.fee
```