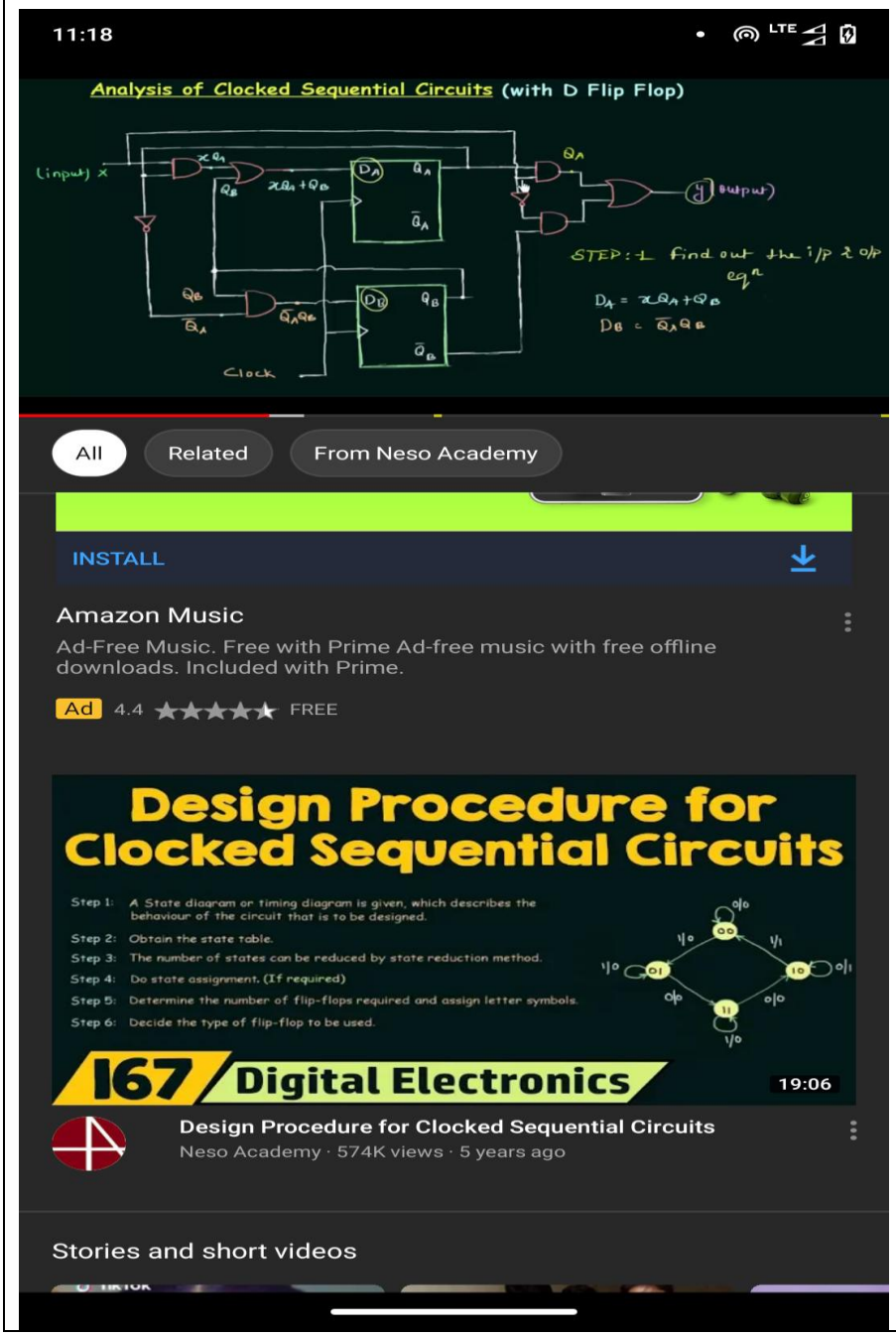
# DAILY ASSESSMENT FORMAT

| | | | |
|---|---|---|---|
| **Date:** | **29May 2020** | **Name:** | **Poojary Sushant** |
| **Course:** | **Logic Design** | **USN:** | **4AL18EC400** |
| **Topic:** | **Analysis of Clocked Sequential circuits,Digital Clock Design** | **Semester & Section:** | **6th sem 'B"** |
| **Github Repository:** | **Sushant7026** | | |

## FORENOON SESSION DETAILS

**Image of session**



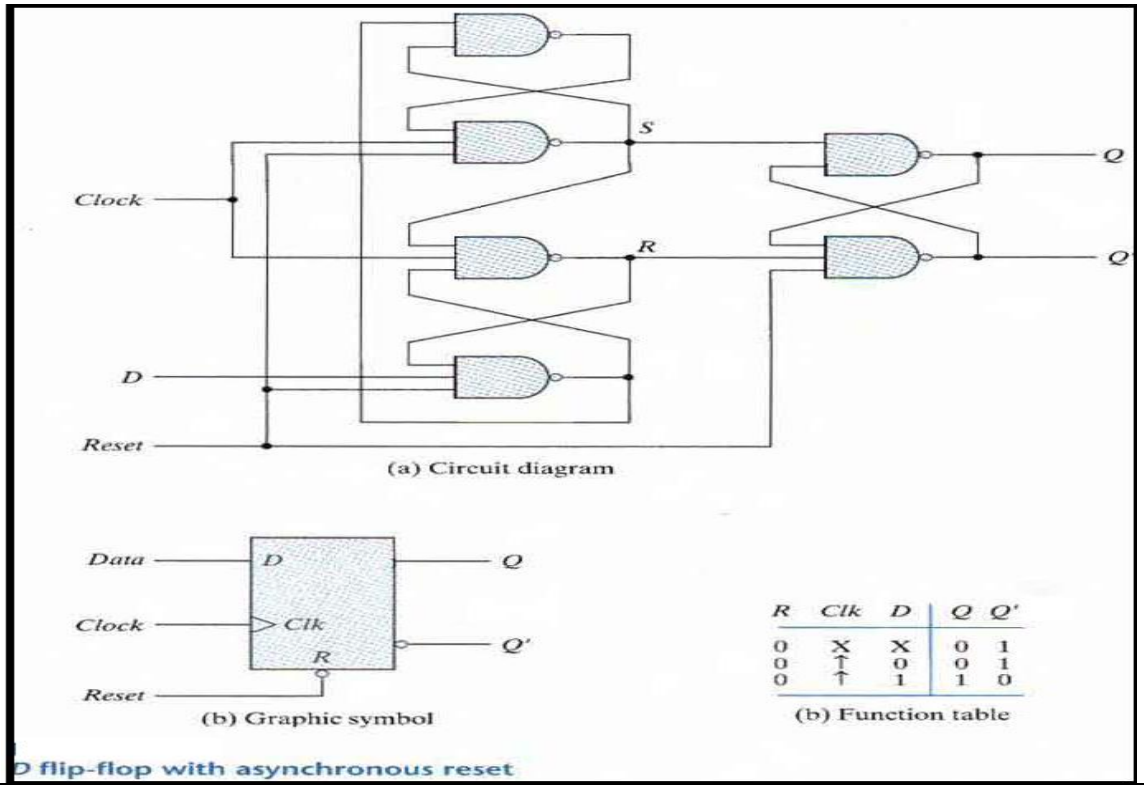
**Report – Report can be typed or hand written for up to two pages.**

# 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 information available in a state table can be represented graphically in the form of a **state diagram.** In this type of diagram a state is represented by a circle and the (clock-triggered) transitions between states are indicated by directed lines connecting the circles.
- The time sequence of inputs, outputs, and flip-flop states can be enumerated in a state table (transition table). The table has four parts present state, next state, inputs and outputs.
- In general a sequential circuit with **'m' flip-flops and 'n' inputs needs $2^{m+n}$** rows in the state table.

**Positive Edge Triggered D Flip-flop**

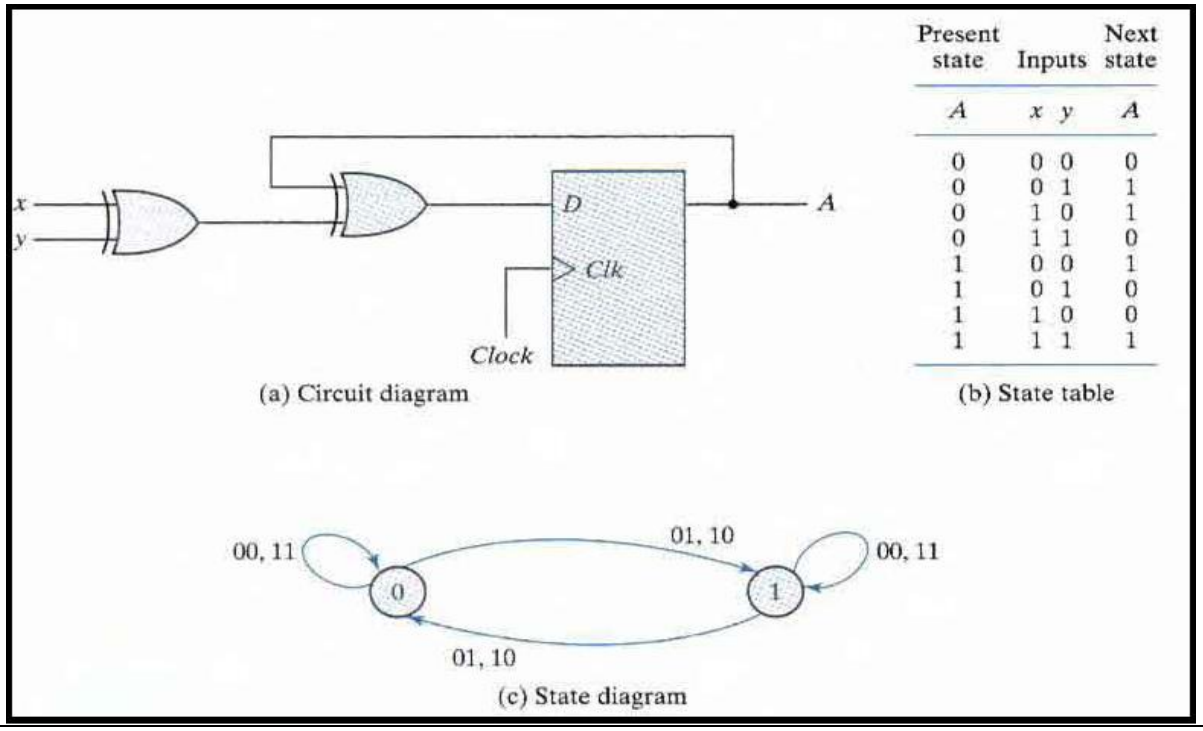
- A circuit diagram of a Positive edge triggered D Flip-flop is shown as below. It has an **additional reset input** connected to the three NAND gates.



- 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.
- Two other connections from the reset input ensure that the S input of the third **SR latch stays at logic 1** while the reset input is at 0 regardless of the values of D and Clk.
- Function table suggests that:
    - **When R = 0, the output is set to 0 (independent of D and Clk).**
    - The clock at Clk is shown with an upward arrow to indicate that the flip-flop triggers on the positive edge of the clock.
    - The value in D is transferred to Q with every positive-edge clock signal provided that R = 1.

## Analysis with D Flip-Flops

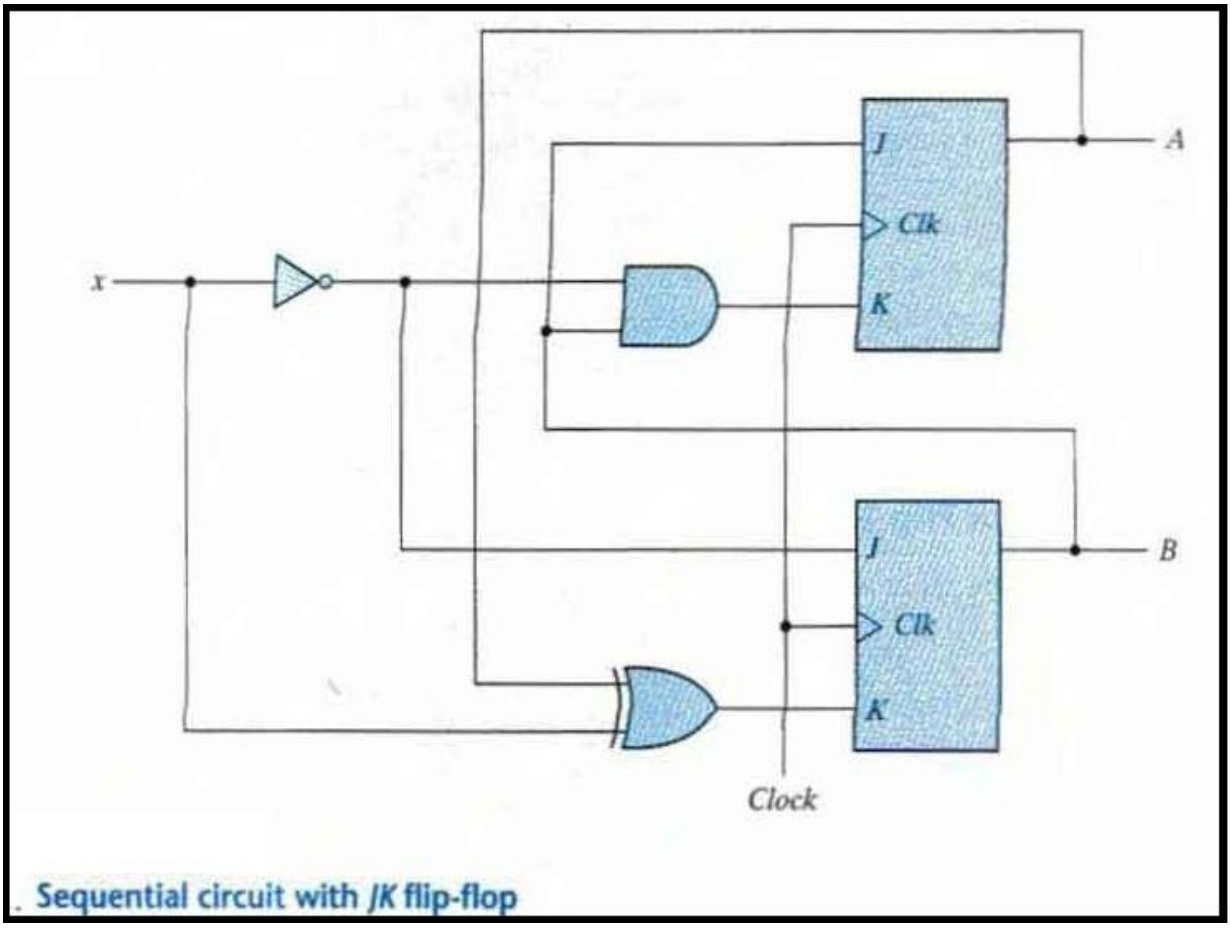
- The input equation of a D Flip-flop is given by $\mathbf{D_A = A \oplus x \oplus y}$. $D_A$ means a D Flip-flop with output A.
- The x and y variables are the inputs to the circuit. No output equations are given, which implies that the output comes from the output of the flip-flop.
- The state table has one column for the present state of flip-flop 'A' two columns for the two inputs, and one column for the next state of A.
- The next-state values are obtained from the state equation $\mathbf{A(t + 1) = A \oplus x \oplus y}$.
- The expression specifies an odd function and is equal to 1 when only one variable is 1 or when all three variables are 1.



## Analysis with JK Flip-Flops

- The circuit can be specified by the flip-flop input equations:
  - **$J_A = B$; $K_A = Bx'$**
  - **$J_B = x'$; $K_B = A'x + Ax' = A \oplus x$**

- The next state of each flip-flop is evaluated from the corresponding J and K inputs and the characteristic table of the JK flip-flop listed as:
  - **When J = 1 and K = 0 the next state is 1**
  - **When J = 0 and K = 1 the next state is 0**
  - **When J = 0 and K = 1 there is no change of state and the next-state value is the same as that of the present state.**
  - **When J = K = 1, the next-state bit is the complement of the present-state bit.**



Sequential circuit with *JK* flip-flop

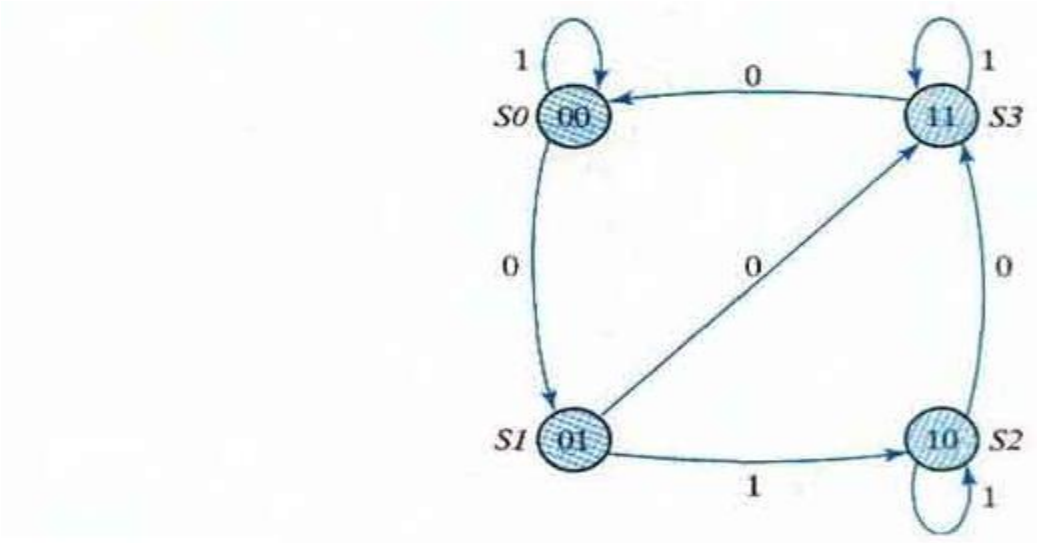
- The characteristic equations for the flip-flops are
  - **$A(t + 1) = JA' + K'A$**
  - **$B(t + 1) = JB' + K'B$**

- This gives us the state equation of A by substituting the values of $J_A$, $K_A$
    - **A(t + 1) = BA' + (Bx')'A = A'B + AB' + Ax**

*State Table for Sequential Circuit with JK Flip-Flops*

| Present State | | Input | Next State | | Flip-Flop Inputs | | | |
|---|---|---|---|---|---|---|---|---|
| A | B | x | A | B | $J_A$ | $K_A$ | $J_B$ | $K_B$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

- The state equation provides the bit values for the column headed "Next State" for A in the state table. Similarly, the state equation for flip-flop B can be derived from the characteristic equation by substituting the values of $J_B$ and $K_B$.:
    - **B(t + 1) = x'B' + (A ⊕ x)'B = B'x' + ABx + A'Bx'**



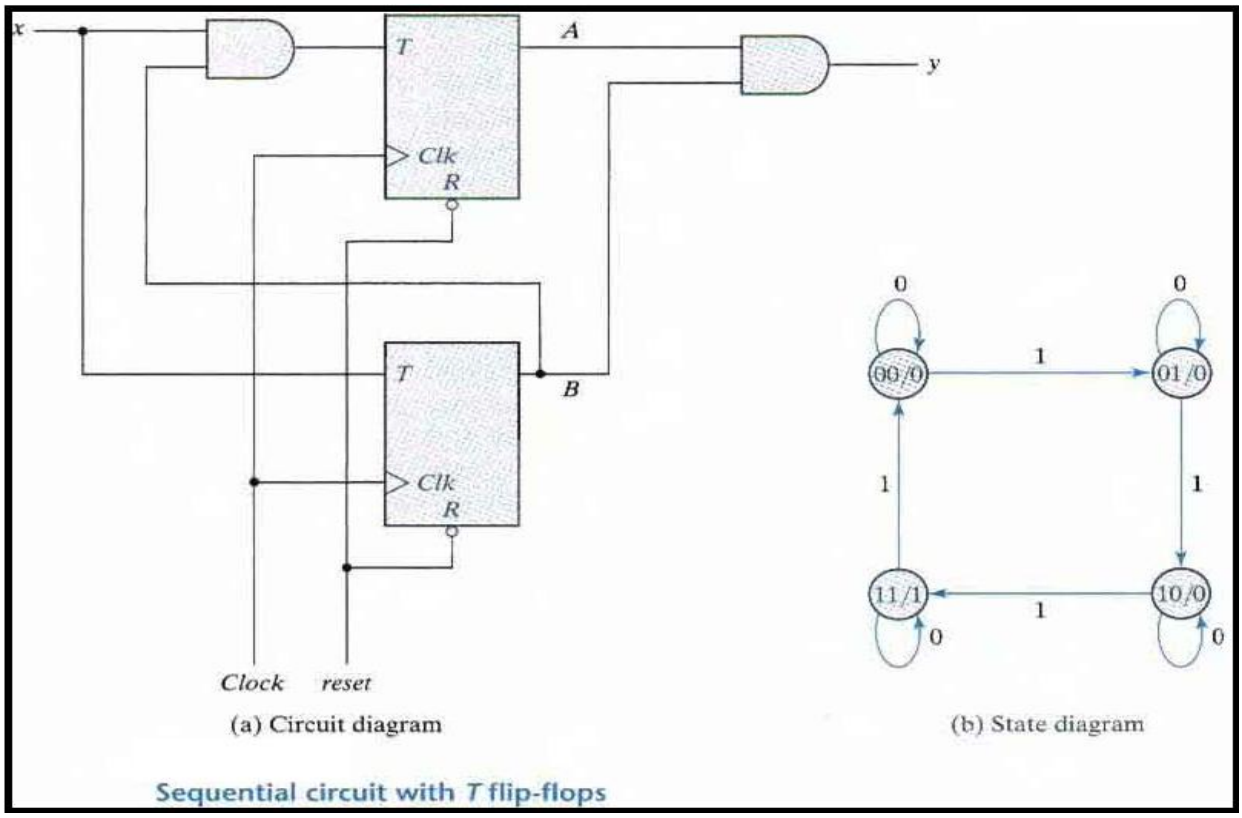

State diagram of the circuit

## Analysis with T Flip-Flops

- The circuit can be specified by the characteristic equations:
  - **Q(t+1) = T ⊕ Q = T'Q + TQ'**

- The sequential circuit has two flip-flops A and B, one input x, and one output y and can be described algebraically by two input equations and an output equation:
  - **$T_A$ = Bx**
  - **$T_B$ = x**
  - **y = AB**

- The state table for the circuit is listed below. The values for y are obtained from the output equation. The values for the next state can be derived from the state equations by substituting $T_A$ and $T_B$ in the characteristic equations yielding:
  - **A(t + 1) = (Bx)' A + (Bx)A' = AB' + Ax' + A'Bx**
  - **B(t + 1) = x ⊕ B**



Sequential circuit with *T* flip-flops

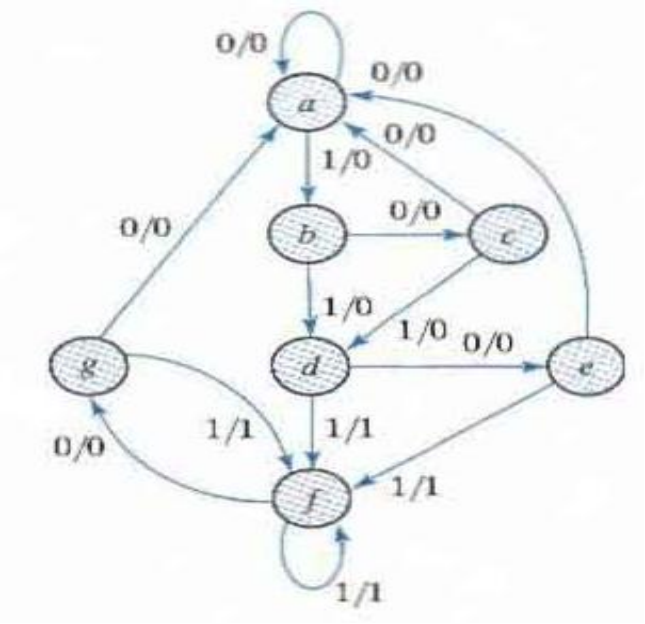
*State Table for Sequential Circuit with T Flip-Flops*

| Present State | | Input | Next State | | Output |
|---|---|---|---|---|---|
| A | B | x | A | B | y |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |

## STATE REDUCTION AND ASSIGNMENT

- Two sequential circuits may exhibit the same input-output behavior but have a different number of internal states in their state diagram.
- Certain properties of sequential circuits may simplify a design by reducing the number of gates and flip-flops it uses. Reducing the number of flip-flops reduces the cost of a circuit.
- The reduction in the number of flip-flops in a sequential circuit is referred to as the state reduction problem. State-reduction algorithms are concerned with procedures for reducing the number of states in a state table while keeping the external input-output requirements unchanged

**Example of State Reduction**




- First we need the state table: it is more convenient to apply procedures for state reduction with the use of a table rather than a diagram

*State Table*

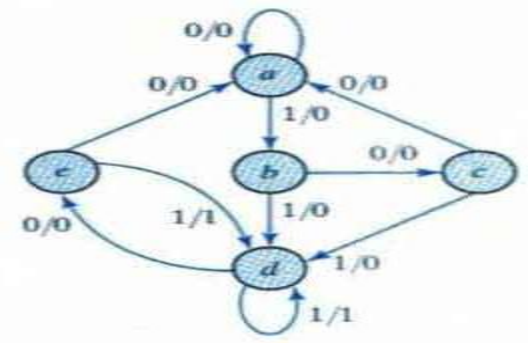
| Present State | Next State | | Output | |
|---|---|---|---|---|
| | $x = 0$ | $x = 1$ | $x = 0$ | $x = 1$ |
| $a$ | $a$ | $b$ | 0 | 0 |
| $b$ | $c$ | $d$ | 0 | 0 |
| $c$ | $a$ | $d$ | 0 | 0 |
| $d$ | $e$ | $f$ | 0 | 1 |
| $e$ | $a$ | $f$ | 0 | 1 |
| $f$ | $g$ | $f$ | 0 | 1 |
| $g$ | $a$ | $f$ | 0 | 1 |

- Then we apply the reduction algorithms *"Two states are said to be equivalent if for each member of the set of inputs they give exactly the same output and send the circuit either to the same state or to an equivalent state."*
- When two states are equivalent one of them can be removed without altering the input-output relationships.
- Going through the state table, we look for two present states that go to the same next state and have the same output for both input combinations. States **g and e** are two such states.
- The procedure of removing a state and replacing it by its equivalent is *"The row with present state g is removed and state g is replaced by state e each time it occurs in the columns headed "NextState,"*

*Reduced State Table*

| Present State | Next State | | Output | |
|---|---|---|---|---|
| | $x = 0$ | $x = 1$ | $x = 0$ | $x = 1$ |
| $a$ | $a$ | $b$ | 0 | 0 |
| $b$ | $c$ | $d$ | 0 | 0 |
| $c$ | $a$ | $d$ | 0 | 0 |
| $d$ | $e$ | $d$ | 0 | 1 |
| $e$ | $a$ | $d$ | 0 | 1 |

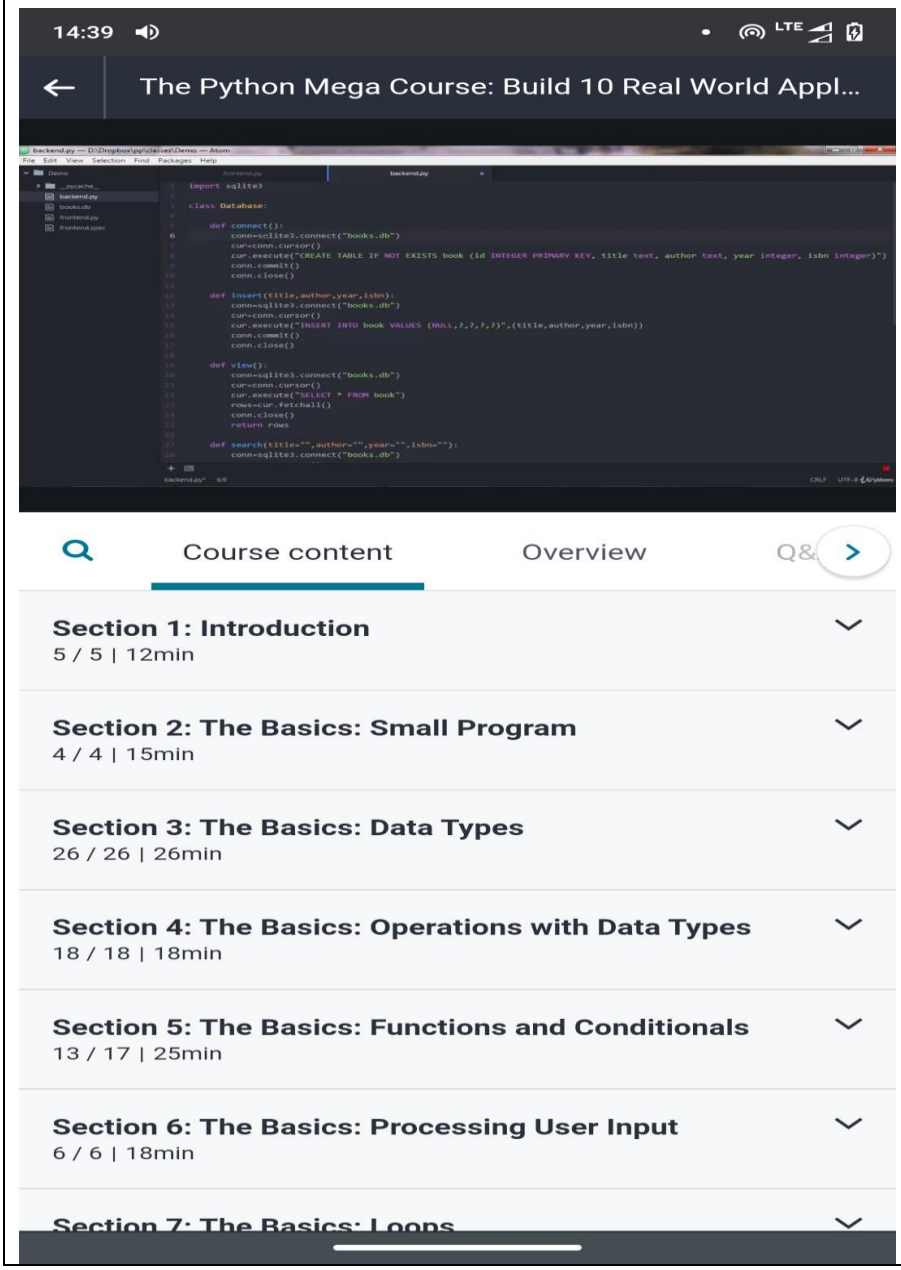
- In general reducing the number of states in a state table may result in a circuit with less equipents. But it does not guarantee a saving in the number of flip-flops or the number of gates.



| | | | |
|---|---|---|---|
| **Date:** | **May 2020** | **Name:** | **Poojary Sushant** |
| **Course:** | **Python On Udemy** | **USN:** | **4AL18EC400** |
| **Topic:** | **Object Oriented Programming** | **Semester & Section:** | **6th & 'B,** |
| **GitHub Repository :** | **Sushant7026** | | |

**AFTERNOON SESSION DETAILS**

**Image of session**



GUI in OOP Design:
Frontend.py

```
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])
    e3.delete(0,END)
    e3.insert(END,selected_tuple[3])
    e4.delete(0,END)
    e4.insert(END,selected_tuple[4])

def view_command():
    list1.delete(0,END)
    for row in database.view():
        list1.insert(END,row)

def search_command():
    list1.delete(0,END)
    for row in database.search(title_text.get(),author_text.get(),year_text.get(),isbn_text.get()):
        list1.insert(END,row)

def add_command():
    database.insert(title_text.get(),author_text.get(),year_text.get(),isbn_text.get())
    list1.delete(0,END)
    list1.insert(END,(title_text.get(),author_text.get(),year_text.get(),isbn_text.get()))

def delete_command():
    database.delete(selected_tuple[0])

def update_command():
    database.update(selected_tuple[0],title_text.get(),author_text.get(),year_text.get(),isbn_text.get())

window=Tk()

window.wm_title("BookStore")

l1=Label(window,text="Title")
l1.grid(row=0,column=0)
```

```
l2=Label(window,text="Author")
l2.grid(row=0,column=2)

l3=Label(window,text="Year")
l3.grid(row=1,column=0)

l4=Label(window,text="ISBN")
l4.grid(row=1,column=2)

title_text=StringVar()
e1=Entry(window,textvariable=title_text)
e1.grid(row=0,column=1)

author_text=StringVar()
e2=Entry(window,textvariable=author_text)
e2.grid(row=0,column=3)

year_text=StringVar()
e3=Entry(window,textvariable=year_text)
e3.grid(row=1,column=1)

isbn_text=StringVar()
e4=Entry(window,textvariable=isbn_text)
e4.grid(row=1,column=3)

list1=Listbox(window, height=6,width=35)
list1.grid(row=2,column=0,rowspan=6,columnspan=2)

sb1=Scrollbar(window)
sb1.grid(row=2,column=2,rowspan=6)

list1.configure(yscrollcommand=sb1.set)
sb1.configure(command=list1.yview)

list1.bind('<<ListboxSelect>>',get_selected_row)

b1=Button(window,text="View all", width=12,command=view_command)
b1.grid(row=2,column=3)

b2=Button(window,text="Search entry", width=12,command=search_command)
b2.grid(row=3,column=3)

b3=Button(window,text="Add entry", width=12,command=add_command)
b3.grid(row=4,column=3)

b4=Button(window,text="Update selected", width=12,command=update_command)
```

```
b4.grid(row=5,column=3)

b5=Button(window,text="Delete selected", width=12,command=delete_command)
b5.grid(row=6,column=3)

b6=Button(window,text="Close", width=12,command=window.destroy)
b6.grid(row=7,column=3)
window.mainloop()
```

Backend.py

```
import sqlite3

class Database:

    def __init__(self, db):
        self.conn=sqlite3.connect(db)
        self.cur=self.conn.cursor()
        self.cur.execute("CREATE TABLE IF NOT EXISTS book (id INTEGER PRIMARY KEY, title text, author text,
year integer, isbn integer)")
        self.conn.commit()

    def insert(self,title,author,year,isbn):
        self.cur.execute("INSERT INTO book VALUES (NULL,?,?,?,?)",(title,author,year,isbn))
        self.conn.commit()

    def view(self):
        self.cur.execute("SELECT * FROM book")
        rows=self.cur.fetchall()
        return rows

    def search(self,title="",author="",year="",isbn=""):
        self.cur.execute("SELECT * FROM book WHERE title=? OR author=? OR year=? OR isbn=?", (title,author,year,
isbn))
        rows=self.cur.fetchall()
        return rows

    def delete(self,id):
        self.cur.execute("DELETE FROM book WHERE id=?",(id,))
        self.conn.commit()

    def update(self,id,title,author,year,isbn):
        self.cur.execute("UPDATE book SET title=?, author=?, year=?, isbn=? WHERE id=?",(title,author,year,isbn,id))
        self.conn.commit()

    def __del__(self):
        self.conn.close()
```

## Report of Webinar on Preparing for Next Normal: By Mr. Mohan Kumar

- Case Study
- Crisis
  - ❑ Danger
  - ❑ Opportunity
- 61% of Enterprise Line of Business

Challenges During Times of Disruption

- Organisational Barriers
- Shifting Customers Emotions

Business Impact:

- Spotting Disruptive Business and People Early
- The Urgency of Innovations
- Continuous Evaluation of Your Social Media Strategy
- Continuous Education Critical
- Technicians
- Customer Experience as a new Battlefield
- Challenging your Economic Model

Next Normal
React ,Reimage and Realign

How Organisations React to the Covid 19?
Trust->Compassion->Stability->Hope
"We cannot solve problems in same Thinking "-Einstein

- Resilient Dynamism
- Digital Transmission
- Economic Crisis

Digital Transmission in Education

- Institution
- Forcefully

Exhibit->Design->Thinking->Innovation.

Skills:

- Niche
- Markable

Commodify

```
# CODE CHALLENGE
"""code challenge

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1QaYCuZkl55rOzD2SqeiPqpkU2fZ4a1S6
"""


import pandas as pd
from google.colab import files
uploaded = files.upload()

import io
df = pd.read_csv(io.BytesIO(uploaded['a01_3.csv']))
df

import numpy as np
import matplotlib.pyplot as plt
import csv
x = []
y = []

with open('a01_3.csv','r') as csvfile:
    plots = csv.reader(csvfile, delimiter=',')
    for row in plots:
        x.append(row[0])
        y.append(row[1])


plt.plot(x,y)
plt.xlim(50,100)
#plt.ylim(0,40)
plt.xlabel('Time')
plt.ylabel('Raw_ECG')
plt.title('ECG Signal')
```