**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **29/5/2020** | **Name:** | **CHESMI B R** |
| **Course:** | **Logic design** | **USN:** | **4AL16EC100** |
| **Topic:** | **Analysis of clocked sequential circuits, Digital clock design.** | **Semester & Section:** | **8th sem A sec** |
| **Github Repository:** | **chesmibr** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report –**  A Sequential circuit combinational logic circuit that consists of inputs variable (X), logic gates (Computational circuit), and output variable (Z).  Combinational circuit produces an output based on input variable only, but Sequential circuit produces an output based on current input and previous input variables. That means sequential circuits include memory elements which are capable of storing binary information. That binary information defines the state of the sequential circuit at that time. A latch capable of storing one bit of information.  There are two types of input to the combinational logic :   * External inputs which not controlled by the circuit. * Internal inputs which are a function of a previous output states.   Secondary inputs are state variables produced by the storage elements, where as secondary outputs are excitations for the storage elements.  **Types of Sequential Circuits :**  There are two types of sequential circuit :   * Asynchronous sequential circuit – These circuit do not use a clock signal but uses the pulses of the inputs. These circuits are faster than synchronous sequential circuits because there is clock pulse and change their state immediately when there is a change in the input signal. We use asynchronous sequential circuits when speed of operation is important and independent of internal clock pulse.   But these circuits are more difficult to design and their output is uncertain.   * Synchronous sequential circuit – These circuit uses clock signal and level inputs (or pulsed) (with restrictions on pulse width and circuit propagation). The output pulse is the same duration as the clock pulse for the clocked sequential circuits. Since they wait for the next clock pulse to arrive to perform the next operation, so these circuits are bit slower compared to asynchronous. Level output changes state at the start of an input pulse and remains in that until the next input or clock pulse.   We use synchronous sequential circuit in synchronous counters, flip flops, and in the design of MOORE-MEALY state management machines.  We use sequential circuits to design Counters, Registers, RAM, MOORE/MEALY Machine and other state retaining machines.  The behavior of a clocked sequential circuit is determined from   * The inputs * The outputs * The state of its flip-flops   The outputs and the next state are both a function of the inputs and the present state .To analyze a sequential circuit, we can use State equations, State table, State diagram, and Flip-Flop input equations. **Digital Clock Circuit Design Using 7493** The 4 blocks of a digital clock are   * 1 Hz clock generator to generate 1 PPS (pulse per second) signal to the seconds block. * SECONDS block - contains a divide by 10 circuit followed by a divide by 6 circuit. Will generate a 1 PPM (pulse per minute) signal to the minutes block. The [BCD](http://electronics-course.com/number-systems) outputs connect to the [BCD to Seven Segment circuit](http://electronics-course.com/bcd-7-segment) to display the seconds values. * MINUTES block - identical to the seconds block it contains 2 dividers; a divide by 10 followed by a divide by 6. Will generate a 1 PPH (pulse per hour) signal to the HOURS block. The BCD outputs connects to the BCD to Seven Segment circuit to display the minutes values. * HOURS block - depending on whether it is a 12 or 24H clock, will have a divide 24 or divide by 12. For 24H, it will count from 00 to 23. For 12H, it will count from 00 to 11. The BCD outputs connects to the BCD to Seven Segment circuit to display the hours values.   The clock can be designed as a 24H or 12H clock. We will explain the steps to arrive at the combinational logic to obtain a 12H clock and we will leave it to the reader to design the 24H clock as an exercise. Click [hints](http://electronics-course.com/digital-clock#hide) if you need help to design the 24H clock.  12H Clock:   * In order to use all 4 bits of the IC1 (ones) counter, Q0 must be connected to CP1. Q0 is LSB and Q3 is MSB. The input clock is connected to CP0. * Since less than 3 bits are required for IC2 (tens), Q0 is not used. Q1 is LSB and Q3 is MSB. The input clock is connected to CP1. * The truth table of the counter is abbreviated - omitting those rows where the MR inputs to the counters are 0. Recall that for the 7493, a 1 to the MR will reset the counters to 0. * To simplify the table, K is Q0 of IC1 (ones), G is Q0 of IC2 (tens) and so on. * For the 12H clock, when the count in BCD reaches   + 0A, IC1 must be cleared (Y=1)   + 12, IC1 must be cleared (Y=1) and IC2 must be cleared (X=1) * Using [SOP (sum of products)](http://electronics-course.com/sum-of-products), we obtain   + Y = HJ + GJ where Y is the IC1 MR1, MR2 inputs connected together   + X = GJ where X is the IC2 MR1, MR2 inputs connected together  Limitations:  * The clock cannot be set to the correct time. Hint - use additional logic to allow the 1 PPS clock to drive the MINUTES and HOURS block depending on a button press. Below is the block diagram of one solution using a [2 to 1 multiplexer](http://electronics-course.com/mux#tt2). Depending on SET, either the 1 PPS (Pulse Per Second) or the 1 PPH (Pulse Per Hour) clock drives the Hour circuit.   The 12H clock counts from 00 to 11 rather than 01 to 12. Hint - use regular JK flip flops (74LS73) instead of the 74LS93 so on terminal count, the counter output is preset to 01. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:29/5/2020** |  | **Name: Chesmi** |  | |
| **Course: Python** |  | **USN:4AL16EC100** |  | |
| **Topic: object oriented programming.** |  | **Sem :8th sem**  **Section: A sec** |  | |
| **AFTERNOON SESSION DETAILS** | | | |
| **SESSION IMAGE :**        **REPORT:**  These are the frontend.py and backend.py scripts in OOP style. To execute this program you should execute the frontend.py file.  #frontend.py  from tkinter import \*  from backend import Database  database=Database("books.db"  class Window(object):  def \_\_init\_\_(self,window):  self.window = window  self.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)  self.title\_text=StringVar()  self.e1=Entry(window,textvariable=self.title\_text)  self.e1.grid(row=0,column=1  self.author\_text=StringVar()  self.e2=Entry(window,textvariable=self.author\_text)  self.e2.grid(row=0,column=3)  self.year\_text=StringVar()  self.e3=Entry(window,textvariable=self.year\_text)  self.e3.grid(row=1,column=1.  self.isbn\_text=StringVar()  self.e4=Entry(window,textvariable=self.isbn\_text)  self.e4.grid(row=1,column=3)  self.list1=Listbox(window, height=6,width=35)  self.list1.grid(row=2,column=0,rowspan=6,columnspan=2)  sb1=Scrollbar(window)  sb1.grid(row=2,column=2,rowspan=6)  self.list1.configure(yscrollcommand=sb1.set)  sb1.configure(command=self.list1.yview)  self.list1.bind('<<ListboxSelect>>',self.get\_selected\_row)  b1=Button(window,text="View all", width=12,command=self.view\_command)  b1.grid(row=2,column=3)  b2=Button(window,text="Search entry", width=12,command=self.search\_command)  b2.grid(row=3,column=3)  b3=Button(window,text="Add entry", width=12,command=self.add\_command)  b3.grid(row=4,column=3)  b4=Button(window,text="Update selected", width=12,command=self.update\_command)  b4.grid(row=5,column=3)  b5=Button(window,text="Delete selected", width=12,command=self.delete\_command)  b5.grid(row=6,column=3)  b6=Button(window,text="Close", width=12,command=window.destroy)  b6.grid(row=7,column=3)  def get\_selected\_row(self,event):  index=self.list1.curselection()[0]  self.selected\_tuple=self.list1.get(index)  self.e1.delete(0,END)  self.e1.insert(END,self.selected\_tuple[1])  self.e2.delete(0,END)  self.e2.insert(END,self.selected\_tuple[2])  self.e3.delete(0,END)  self.e3.insert(END,self.selected\_tuple[3])  self.e4.delete(0,END)  self.e4.insert(END,self.selected\_tuple[4])  def view\_command(self):  self.list1.delete(0,END)  for row in database.view():  self.list1.insert(END,row  def search\_command(self):  self.list1.delete(0,END)  for row in database.search(self.title\_text.get(),self.author\_text.get(),self.year\_text.get(),self.isbn\_text.get()):  self.list1.insert(END,row.  def add\_command(self):  database.insert(self.title\_text.get(),self.author\_text.get(),self.year\_text.get(),self.isbn\_text.get())  self.list1.delete(0,END)  self.list1.insert(END,(self.title\_text.get(),self.author\_text.get(),self.year\_text.get(),self.isbn\_text.get()))  def delete\_command(self):  database.delete(self.selected\_tuple[0])  def update\_command(self):  database.update(self.selected\_tuple[0],self.title\_text.get(),self.author\_text.get(),self.year\_text.get(),self.isbn\_text.get())  window=Tk()  Window(window)  window.mainloop()  And below you will also find the backend.py script in OOP:   1. #backend.py 2. import sqlite3 3. class Database: 4. def \_\_init\_\_(self, db): 5. self.conn=sqlite3.connect(db) 6. self.cur=self.conn.cursor() 7. self.cur.execute("CREATE TABLE IF NOT EXISTS book (id INTEGER PRIMARY KEY, title text, author text, year integer, isbn integer)") 8. self.conn.commit() 9. def insert(self,title,author,year,isbn): 10. self.cur.execute("INSERT INTO book VALUES (NULL,?,?,?,?)",(title,author,year,isbn)) 11. self.conn.commit() 12. def view(self): 13. self.cur.execute("SELECT \* FROM book") 14. rows=self.cur.fetchall() 15. return rows 16. def search(self,title="",author="",year="",isbn=""): 17. self.cur.execute("SELECT \* FROM book WHERE title=? OR author=? OR year=? OR isbn=?", (title,author,year,isbn)) 18. rows=self.cur.fetchall() 19. return rows 20. def delete(self,id): 21. self.cur.execute("DELETE FROM book WHERE id=?",(id,)) 22. self.conn.commit() 23. def update(self,id,title,author,year,isbn): 24. self.cur.execute("UPDATE book SET title=?, author=?, year=?, isbn=? WHERE id=?",(title,author,year,isbn,id)) 25. self.conn.commit() 26. def \_\_del\_\_(self): 27. self.conn.close()   **SECOND CODE CHALLENGE**   |  | | --- | | x=0.01:0.01:2; | |  | default=input('Press 1 if u want default ecg signal else press 2:\n'); | |  | if(default==1) | |  | li=30/72; | |  |  | |  | a\_pwav=0.25; | |  | d\_pwav=0.09; | |  | t\_pwav=0.16; | |  |  | |  | a\_qwav=0.025; | |  | d\_qwav=0.066; | |  | t\_qwav=0.166; | |  |  | |  | a\_qrswav=1.6; | |  | d\_qrswav=0.11; | |  |  | |  | a\_swav=0.25; | |  | d\_swav=0.066; | |  | t\_swav=0.09; | |  |  | |  | a\_twav=0.35; | |  | d\_twav=0.142; | |  | t\_twav=0.2; | |  |  | |  | a\_uwav=0.035; | |  | d\_uwav=0.0476; | |  | t\_uwav=0.433; | |  | else | |  | rate=input('\n\nenter the heart beat rate :'); | |  | li=30/rate; | |  |  | |  | %p wave specifications | |  | fprintf('\n\np wave specifications\n'); | |  | d=input('Enter 1 for default specification else press 2: \n'); | |  | if(d==1) | |  | a\_pwav=0.25; | |  | d\_pwav=0.09; | |  | t\_pwav=0.16; | |  | else | |  | a\_pwav=input('amplitude = '); | |  | d\_pwav=input('duration = '); | |  | t\_pwav=input('p-r interval = '); | |  | d=0; | |  | end | |  |  | |  |  | |  | %q wave specifications | |  | fprintf('\n\nq wave specifications\n'); | |  | d=input('Enter 1 for default specification else press 2: \n'); | |  | if(d==1) | |  | a\_qwav=0.025; | |  | d\_qwav=0.066; | |  | t\_qwav=0.166; | |  | else | |  | a\_qwav=input('amplitude = '); | |  | d\_qwav=input('duration = '); | |  | t\_qwav=0.166; | |  | d=0; | |  | end | |  |  | |  |  | |  |  | |  | %qrs wave specifications | |  | fprintf('\n\nqrs wave specifications\n'); | |  | d=input('Enter 1 for default specification else press 2: \n'); | |  | if(d==1) | |  | a\_qrswav=1.6; | |  | d\_qrswav=0.11; | |  | else | |  | a\_qrswav=input('amplitude = '); | |  | d\_qrswav=input('duration = '); | |  | d=0; | |  | end | |  |  | |  |  | |  |  | |  | %s wave specifications | |  | fprintf('\n\ns wave specifications\n'); | |  | d=input('Enter 1 for default specification else press 2: \n'); | |  | if(d==1) | |  | a\_swav=0.25; | |  | d\_swav=0.066; | |  | t\_swav=0.09; | |  | else | |  | a\_swav=input('amplitude = '); | |  | d\_swav=input('duration = '); | |  | t\_swav=0.09; | |  | d=0; | |  | end | |  |  | |  |  | |  | %t wave specifications | |  | fprintf('\n\nt wave specifications\n'); | |  | d=input('Enter 1 for default specification else press 2: \n'); | |  | if(d==1) | |  | a\_twav=0.35; | |  | d\_twav=0.142; | |  | t\_twav=0.2; | |  | else | |  | a\_twav=input('amplitude = '); | |  | d\_twav=input('duration = '); | |  | t\_twav=input('s-t interval = '); | |  | d=0; | |  | end | |  |  | |  |  | |  | %u wave specifications | |  | fprintf('\n\nu wave specifications\n'); | |  | d=input('Enter 1 for default specification else press 2: \n'); | |  | if(d==1) | |  | a\_uwav=0.035; | |  | d\_uwav=0.0476; | |  | t\_uwav=0.433; | |  | else | |  | a\_uwav=input('amplitude = '); | |  | d\_uwav=input('duration = '); | |  | t\_uwav=0.433; | |  | d=0; | |  | end | |  |  | |  |  | |  |  | |  | end | |  | pwav=p\_wav(x,a\_pwav,d\_pwav,t\_pwav,li); | |  |  | |  | %qwav output | |  | qwav=q\_wav(x,a\_qwav,d\_qwav,t\_qwav,li); | |  |  | |  | %qrswav output | |  | qrswav=qrs\_wav(x,a\_qrswav,d\_qrswav,li); | |  | %swav output | |  | swav=s\_wav(x,a\_swav,d\_swav,t\_swav,li); | |  |  | |  | %twav output | |  | twav=t\_wav(x,a\_twav,d\_twav,t\_twav,li); | |  |  | |  | %uwav output | |  | uwav=u\_wav(x,a\_uwav,d\_uwav,t\_uwav,li); | |  | %ecg output | |  | ecg=pwav+qrswav+twav+swav+qwav+uwav; | |  | figure(1) | |  | plot(x,ecg); | | | | |