**DAILY ASSESSMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **29/05/2020** | **Name:** | **Dhavala** |
| **Course:** | **Logic Design** | **USN:** | **4AL17EC027** |
| **Topic:** | * **Analysis of clocked sequential circuits** * **Digital clock design** | **Semester & Section:** | **6TH SEM & A Section** |
| **Github Repository:** | **Dhavala27** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
|  |

|  |
| --- |
| **Report**  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-flops is described algebraically by a set of Boolean func­tions called flip-flop input equations (or excitation equations). * 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 2m+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.   Positive Edge Triggered D Flip-flop   * 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 DA = A ⊕ x ⊕ y. DA 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 A(t + 1) = A ⊕ x ⊕ 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 D Flip-Flops  Analysis with JK Flip-Flops   * The circuit can be specified by the flip-flop input equations:   + JA = B; KA = Bx'   + JB = x'; KB = A'x + Ax' = A ⊕ 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 = 0 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.   Analysis with jk Flip-Flops   * 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 JA, KA   Analysis with jk Flip-Flops   * + A(t + 1) = BA' + (Bx')'A = A'B + AB' + Ax * 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 JB and KB.:   + B(t + 1) = x'B' + (A ⊕ x)'B = B'x' + ABx + A'Bx'   Analysis with jk Flip-Flops  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:   + TA = Bx   + TB = 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 TA and TB in the characteristic equations yielding:   + A(t + 1) = (Bx)' A + (Bx)A' = AB' + Ax' + A'Bx   + B(t + 1) = x ⊕ B   Analysis with T Flip-Flops  Analysis with T Flip-Flops  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**  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.   Example of State Reduction   * 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 "Next State,"* * Similarly, states f and d are equivalent, and state f can be removed and replaced by d.   Example of State Reduction   * In general reducing the number of states in a state table may result in a circuit with less equipment’s. But it does not guarantee a saving in the number of flip-flops or the number of gates.   Example of State Reduction |

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **29/05/2020** | **Name:** | **Dhavala** |
| **Course:** | **WEBINAR** | **USN:** | **4AL17EC027** |
| **Topic:** | * **Preparation for the next normal** | **Semester & Section:** | **6TH SEM & A Section** |
| **Github Repository:** | **Dhavala27** |  |  |

****

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **29/05/2020** | **Name:** | **Dhavala** |
| **Course:** | **PYTHON** | **USN:** | **4AL17EC027** |
| **Topic:** | * **Object Oriented Programming** | **Semester & Section:** | **6TH SEM & A Section** |
| **Github Repository:** | **Dhavala27** |  |  |

|  |
| --- |
| **AFTERNOON SESSION DETAILS** |
|  |

|  |
| --- |
| **Report** |
| GUI in OOP Design  Alter the frontend.py script containing the GUI code by changing its functional-oriented design into an OOP design.  For your convenience, the files *frontend.py*, *backend.py* (in OOP style), and the *book.db* files are attached in this article's resources.  Solution  Here are the *frontend.py*and *backend.py*scripts in OOP style. To execute this program you should execute the *frontend.py* file.   1. #frontend.py 2. from tkinter import \* 3. from backend import Database 5. database=Database("books.db") 7. class Window(object): 9. def \_\_init\_\_(self,window): 11. self.window = window 13. self.window.wm\_title("BookStore") 15. l1=Label(window,text="Title") 16. l1.grid(row=0,column=0) 18. l2=Label(window,text="Author") 19. l2.grid(row=0,column=2) 21. l3=Label(window,text="Year") 22. l3.grid(row=1,column=0) 24. l4=Label(window,text="ISBN") 25. l4.grid(row=1,column=2) 27. self.title\_text=StringVar() 28. self.e1=Entry(window,textvariable=self.title\_text) 29. self.e1.grid(row=0,column=1) 31. self.author\_text=StringVar() 32. self.e2=Entry(window,textvariable=self.author\_text) 33. self.e2.grid(row=0,column=3) 35. self.year\_text=StringVar() 36. self.e3=Entry(window,textvariable=self.year\_text) 37. self.e3.grid(row=1,column=1) 39. self.isbn\_text=StringVar() 40. self.e4=Entry(window,textvariable=self.isbn\_text) 41. self.e4.grid(row=1,column=3) 43. self.list1=Listbox(window, height=6,width=35) 44. self.list1.grid(row=2,column=0,rowspan=6,columnspan=2) 46. sb1=Scrollbar(window) 47. sb1.grid(row=2,column=2,rowspan=6) 49. self.list1.configure(yscrollcommand=sb1.set) 50. sb1.configure(command=self.list1.yview) 52. self.list1.bind('<<ListboxSelect>>',self.get\_selected\_row) 54. b1=Button(window,text="View all", width=12,command=self.view\_command) 55. b1.grid(row=2,column=3) 57. b2=Button(window,text="Search entry", width=12,command=self.search\_command) 58. b2.grid(row=3,column=3) 60. b3=Button(window,text="Add entry", width=12,command=self.add\_command) 61. b3.grid(row=4,column=3) 63. b4=Button(window,text="Update selected", width=12,command=self.update\_command) 64. b4.grid(row=5,column=3) 66. b5=Button(window,text="Delete selected", width=12,command=self.delete\_command) 67. b5.grid(row=6,column=3) 69. b6=Button(window,text="Close", width=12,command=window.destroy) 70. b6.grid(row=7,column=3) 72. def get\_selected\_row(self,event): 73. index=self.list1.curselection()[0] 74. self.selected\_tuple=self.list1.get(index) 75. self.e1.delete(0,END) 76. self.e1.insert(END,self.selected\_tuple[1]) 77. self.e2.delete(0,END) 78. self.e2.insert(END,self.selected\_tuple[2]) 79. self.e3.delete(0,END) 80. self.e3.insert(END,self.selected\_tuple[3]) 81. self.e4.delete(0,END) 82. self.e4.insert(END,self.selected\_tuple[4]) 84. def view\_command(self): 85. self.list1.delete(0,END) 86. for row in database.view(): 87. self.list1.insert(END,row) 89. def search\_command(self): 90. self.list1.delete(0,END) 91. for row in database.search(self.title\_text.get(),self.author\_text.get(),self.year\_text.get(),self.isbn\_text.get()): 92. self.list1.insert(END,row) 94. def add\_command(self): 95. database.insert(self.title\_text.get(),self.author\_text.get(),self.year\_text.get(),self.isbn\_text.get()) 96. self.list1.delete(0,END) 97. self.list1.insert(END,(self.title\_text.get(),self.author\_text.get(),self.year\_text.get(),self.isbn\_text.get())) 99. def delete\_command(self): 100. database.delete(self.selected\_tuple[0]) 102. def update\_command(self): 103. database.update(self.selected\_tuple[0],self.title\_text.get(),self.author\_text.get(),self.year\_text.get(),self.isbn\_text.get()) 105. window=Tk() 106. Window(window)   window.mainloop()  And below you will also find the backend.py script in OOP:  #backend.py   1. import sqlite3 2. class Database: 3. def \_\_init\_\_(self, db): 4. self.conn=sqlite3.connect(db) 5. self.cur=self.conn.cursor() 6. self.cur.execute("CREATE TABLE IF NOT EXISTS book (id INTEGER PRIMARY KEY, title text, author text, year integer, isbn integer)") 7. self.conn.commit() 8. def insert(self,title,author,year,isbn): 9. self.cur.execute("INSERT INTO book VALUES (NULL,?,?,?,?)",(title,author,year,isbn)) 10. self.conn.commit() 11. def view(self): 12. self.cur.execute("SELECT \* FROM book") 13. rows=self.cur.fetchall() 14. return rows 15. def search(self,title="",author="",year="",isbn=""): 16. self.cur.execute("SELECT \* FROM book WHERE title=? OR author=? OR year=? OR isbn=?", (title,author,year,isbn)) 17. rows=self.cur.fetchall() 18. return rows 19. def delete(self,id): 20. self.cur.execute("DELETE FROM book WHERE id=?",(id,)) 21. self.conn.commit() 22. def update(self,id,title,author,year,isbn): 23. self.cur.execute("UPDATE book SET title=?, author=?, year=?, isbn=? WHERE id=?",(title,author,year,isbn,id)) 24. self.conn.commit() 25. def \_\_del\_\_(self): 26. self.conn.close() |

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **28/05/2020** | **Name:** | **Dhavala** |
| **Course:** | **Bonus session** | **USN:** | **4AL17EC027** |
| **Topic:** | **Why you should write your own resume?** | **Semester & Section:** | **6TH SEM & A Section** |
| **Github Repository:** | **Dhavala27** |  |  |



