**28 May 2020**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **28 May 2020** | **Name:** | **Srinidhi J C** |
| **Course:** | **Logic Design** | **USN:** | **4al16ec078** |
| **Topic:** | 1. **Boolean equations for digital circuits. Combinational circuits: Conversion of MUX and Decoders to logic gates.** 2. **Design of 7 segment decoder with common anode display** | **Semester & Section:** | **8th -Sem, B-Sec** |
| **Github Repository:** | **SrinidhiJC078** |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **FORENOON SESSION DETAILS** | | | |
| **Image of session**  **A picture containing clock  Description automatically generatedA screenshot of a cell phone  Description automatically generatedA screenshot of a cell phone  Description automatically generated** | | | |
| **Report – Report can be typed or hand written for up to two pages.**  **Boolean Laws**  There are six types of Boolean Laws.  **Commutative law**  Any binary operation which satisfies the following expression is referred to as commutative operation.  Commutative Law  Commutative law states that changing the sequence of the variables does not have any effect on the output of a logic circuit.  **Associative law**  This law states that the order in which the logic operations are performed is irrelevant as their effect is the same.  Associative Law  **Distributive law**  Distributive law states the following condition.  Distributive Law  **AND law**  These laws use the AND operation. Therefore they are called as **AND** laws.  AND Law  **OR law**  These laws use the OR operation. Therefore they are called as **OR** laws.  OR Law  **INVERSION law**  This law uses the NOT operation. The inversion law states that double inversion of a variable results in the original variable itself.  NOT Law  Combinational circuit is a circuit in which we combine the different gates in the circuit, for example encoder, decoder, multiplexer and demultiplexer. Some of the characteristics of combinational circuits are following −   * The output of combinational circuit at any instant of time, depends only on the levels present at input terminals. * The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit. * A combinational circuit can have an n number of inputs and m number of outputs.   **Multiplexers**  Multiplexer is a special type of combinational circuit. There are n-data inputs, one output and m select inputs with 2m = n. It is a digital circuit which selects one of the n data inputs and routes it to the output. The selection of one of the n inputs is done by the selected inputs. Depending on the digital code applied at the selected inputs, one out of n data sources is selected and transmitted to the single output Y. E is called the strobe or enable input which is useful for the cascading. It is generally an active low terminal that means it will perform the required operation when it is low.  **Block diagram**  Block Diagram of n:1 Multiplexer  Multiplexers come in multiple variations   * 2 : 1 multiplexer * 4 : 1 multiplexer * 16 : 1 multiplexer * 32 : 1 multiplexer   **Block Diagram**  2:1 Multiplexer Block Diagram  **Truth Table**  2:1 Multiplexer Truth Table  **Demultiplexers**  A demultiplexer performs the reverse operation of a multiplexer i.e. it receives one input and distributes it over several outputs. It has only one input, n outputs, m select input. At a time only one output line is selected by the select lines and the input is transmitted to the selected output line. A de-multiplexer is equivalent to a single pole multiple way switch as shown in fig.  Demultiplexers comes in multiple variations.   * 1 : 2 demultiplexer * 1 : 4 demultiplexer * 1 : 16 demultiplexer * 1 : 32 demultiplexer   **Block diagram**  Block Diagram of 1:2 Demultiplexer  **Truth Table**  1:2 Demultiplexer Truth Table  **Decoder**  A decoder is a combinational circuit. It has n input and to a maximum m = 2n outputs. Decoder is identical to a demultiplexer without any data input. It performs operations which are exactly opposite to those of an encoder.  **Block diagram**  Block Diagram of Decoder  Examples of Decoders are following.   * Code converters * BCD to seven segment decoders * Nixie tube decoders * Relay actuator   **2 to 4 Line Decoder**  The block diagram of 2 to 4 line decoder is shown in the fig. A and B are the two inputs where D through D are the four outputs. Truth table explains the operations of a decoder. It shows that each output is 1 for only a specific combination of inputs.  **Block diagram**  Block Diagram of 2 to 4 Decoder  **Truth Table**  Truth Table of 2 to 4 Decoder  **Logic Circuit**  Logic Circuit of 2 to 4 Decoder  **Encoder**  Encoder is a combinational circuit which is designed to perform the inverse operation of the decoder. An encoder has n number of input lines and m number of output lines. An encoder produces an m bit binary code corresponding to the digital input number. The encoder accepts an n input digital word and converts it into an m bit another digital word.  **Block diagram**  Block Diagram of encoder  Examples of Encoders are following.   * Priority encoders * Decimal to BCD encoder * Octal to binary encoder * Hexadecimal to binary encoder   **Priority Encoder**  This is a special type of encoder. Priority is given to the input lines. If two or more input line are 1 at the same time, then the input line with highest priority will be considered. There are four input D0, D1, D2, D3 and two output Y0, Y1. Out of the four input D3 has the highest priority and D0 has the lowest priority. That means if D3 = 1 then Y1 Y1 = 11 irrespective of the other inputs. Similarly if D3 = 0 and D2 = 1 then Y1 Y0 = 10 irrespective of the other inputs.  **Block diagram**  Block Diagram of Priority Encoder  **Truth Table**  Truth Table of Priority Encoder  **Logic Circuit**  Logic Circuit of Priority Encoder  **Seven segment** display is an electronic device which consists of seven Light Emitting Diodes (LEDs) arranged in a some definite pattern (common cathode or common anode type), which is used to display Hexadecimal numerals(in this case decimal numbers,as input is BCD i.e., 0-9).Two types of seven segment LED display:   1. **Common Cathode Type:** In this type of display all cathodes of the seven LEDs are connected together to the ground or -Vcc(hence,common cathode) and LED displays digits when some ‘HIGH’ signal is supplied to the individual anodes. 2. **Common Anode Type:** In this type of display all the anodes of the seven LEDs are connected to battery or +Vcc and LED displays digits when some ‘LOW’ signal is supplied to the individual cathodes.   But, seven segment display does not work by directly supplying voltage to different segments of LEDs. First, our decimal number is changed to its BCD equivalent signal then BCD to seven segment decoder converts that signals to the form which is fed to seven segment display.  This BCD to seven segment decoder has four input lines (A, B, C and D) and 7 output lines (a, b, c, d, e, f and g), this output is given to seven segment LED display which displays the decimal number depending upon inputs.    **Truth Table –** For common cathode type BCD to seven segment decoder: | | | |
| **Date:** | **28 May 2020** | **Name:** | **Srinidhi J C** |
| **Course:** | **Python** | **USN:** | **4al16ec078** |
| **Topic:** | **Object Oriented Programming** | **Semester & Section:** | **8th-Sem, B-Sec** |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session**  **A screenshot of a computer  Description automatically generated** | | | |
| **Report – Report can be typed or hand written for up to two pages.**  **In today’s session these are the programs I have learnt:**  #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:  #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()  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()  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()  #insert("The Sun","John Smith",1918,913123132)  #delete(3)  #update(4,"The moon","John Smooth",1917,99999)  #print(view())  #print(search(author="John Smooth")) | | | |