**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **29-05-2020** | **Name:** | **Karthik J** |
| **Course:** | **Logic Design** | **USN:** | **4AL16EC030** |
| **Topic:** | Analysis of clocked sequential circuits, Digital clock design | **Semester & Section:** | **8TH A** |
| **GitHub Repository:** | Karthik-J |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session**  **REPORT:** Analysis of Clocked Sequential Circuits (with D Flip Flop)  * The behavior of a clocked sequential circuit is determined from its inputs, outputs and state of the flip-flops (i.e., the output of the flip-flops). * The analysis of a clocked sequential circuit consists of obtaining a table of a diagramο of the time sequences of inputs, outputs and states.   Consider the following circuit      The circuit has two inputs A and B, one output Z, The circuit has two flip-flops (different types) with outputs Q0 and Q1 (This implies that there are as many as 4 different states in the circuit, namely Q0Q1 = 00, 01, 11, 10). The circuit output depends on the current state (flip-flop outputs) only.      Another way to illustrate the behavior of a clocked sequential circuit is with a state diagram.    g.PNG  Each “bubble” (state bubble) in the state diagram represents one state of the system. The flip-flop outputs that correspond to that state are labeled inside of theν bubble. Each edge leaving a bubble represents a possible transition to another state onceο the active clock edge arrives. The edges are labeled with the input values that cause the transition to occur.ν In this state diagram, the output values are labeled inside of the state bubbles.ο     * **Digital clock design:** * There are many stuffs on the internet about this project, but we are gonna add something or two. * This circuit was a school project and it was a 24h clock, but i decided to extend it to 12h and 24h with the transferring between them using a switch so you can choose whatever mode you want. * Anyway, i made this circuit just for fun so i just simulated it on proteus, i.e we're not going into pcb design.     The main parts of the circuit are as follows:  1- **Timer 555**: Responsible for generating the clock pulses for the counters, the frequency of the output shoul be 1 hz which means 1 second for each pulse.  2- **Counters**: Responsible for generating the time in BCD (Binary Coded decimal).  3- **Decoders**: Takes the BCD of the counter as input and produces 7 segment output .  4- 7 **segments**: Displays the time, of course  MSB---LSB  0:  0000  1:  0001  2:  0010  3:  0011  4:  0100  5:  0101  6:  0110  7:  0111  8:  1000  9:  1001  0:   0000  Remember that 7490 decade counters respond only to the pulses that go from 1 to 0 and notice that this case only happens in the BCD code above when the output changes from 9 to 0 (the Most significant bit changes from 1 to 0). So, we'll just connect the clock input of the 2nd counter to the most significant bit of the output of the first counter.  The 4th counter will be the same as the second one so we are clocking it using the Most Significant Bit of the output of the previous one.  Again, the 5th counter is the same as the 3rd one and takes its clock from the AND gate.  The 5th and the 6th counters are responsible for hours so they are  limited to 23, and resets themselves to 00 when the 5th counter is 4 and the last one is 2 (24).  This is done using and gate with Q2 (3rd bit) of the 5th counter as one input and Q1 (second bit) of the last counter as the other input, and the output of this AND gate will be connected to both resets of the last 2 counters.  When the last counter is 0(0000) or 1(0001), Q1 which is one of the inputs to the AND gate will be 0 so the output of the AND gate will be zero. when it counts to 2 this bit will be 1 so the output of the and gate will depend on the the other input which is Q2 of the previous counter, and this bit will be zero until it reaches 4 (0100),So, the output of the and gate will be 1 (0--->1) resetting both counters to 00,  Screenshot (34).png |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date:** | | **29-05-2020** | **Name:** | **Karthik J** |  |
| **Course:** | | [Programming with Python: Hands-on Introduction for Beginners](https://www.udemy.com/course/python-programming-beginners/) | **USN:** | **4AL16EC030** |  |
| **Topic:** | |  | **Semester & Section:** | **8th A** |  |
|  | **AFTERNOON SESSION DETAILS** | | | | |
|  | **Image of session** | | | | |
|  | Python - Tuples A tuple is an immutable sequence of Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.  Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also.  For example −  tup1 = ('physics', 'chemistry', 1997, 2000);  tup2 = (1, 2, 3, 4, 5 );  tup3 = "a", "b", "c", "d"; Accessing Values in Tuples To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index.  For example −  tup1 = ('physics', 'chemistry', 1997, 2000);  tup2 = (1, 2, 3, 4, 5, 6, 7 );  print "tup1[0]: ", tup1[0];  print "tup2[1:5]: ", tup2[1:5]; Updating Tuples Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples  Example  tup1 = (12, 34.56);  tup2 = ('abc', 'xyz');  # Following action is not valid for tuples  # tup1[0] = 100;  # So, let's create a new tuple as follows  tup3 = tup1 + tup2;  print tup3; Delete Tuple Elements Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.  To explicitly remove an entire tuple, just use the **del** statement.  example −  tup = ('physics', 'chemistry', 1997, 2000);  print tup;  del tup;  print "After deleting tup : ";  print tup;  This produces the following result. Note an exception raised, this is because after **del tup** tuple does not exist any-more − Basic Tuples Operations Tuples respond to the + and \* operators much like strings; they mean concatenation and repetition here too, except that the result is a new tuple, not a string. Indexing, Slicing, and Matrixes Because tuples are sequences, indexing and slicing work the same way for tuples as they do for strings. Assuming following input −  L = ('spam', 'Spam', 'SPAM!') Python Lists The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.  Creating a list is as simple as putting different comma-separated values between square brackets. For example −  list1 = ['physics', 'chemistry', 1997, 2000];  list2 = [1, 2, 3, 4, 5 ];  list3 = ["a", "b", "c", "d"]  Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on Accessing Values in Lists To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index.  **Example**  list1 = ['physics', 'chemistry', 1997, 2000];  list2 = [1, 2, 3, 4, 5, 6, 7 ];  print "list1[0]: ", list1[0]  print "list2[1:5]: ", list2[1:5] Updating Lists You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and you can add to elements in a list with the append() method.  **Example**  list = ['physics', 'chemistry', 1997, 2000];  print "Value available at index 2 : "  print list[2]  list[2] = 2001;  print "New value available at index 2 : "  print list[2] | | | | |