**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **30-May-2020** | **Name:** | **Raziya Banu** |
| **Course:** | **LD** | **USN:** | **4AL16EC058** |
| **Topic:** | **Analysis of clocked sequential circuits** | **Semester & Section:** | **8th sem & ‘B’ section** |
| **Github Repository:** |  |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report –**  In my first session today I have studied about the Analysis of clocked sequential circuits.  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 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **30-May-2020** | **Name:** | **Raziya Banu** | |
| **Course:** | **Udemy** | **USN:** | **4AL16EC058** | |
| **Topic:** | **Sets in Python** | **Semester & Section:** | **8th sem & ‘B’ section** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| **Python Sets:**In this tutorial, you'll learn everything about Python sets; how they are created, adding or removing elements from them, and all operations performed on sets in Python. A set is an unordered collection of items. Every set element is unique (no duplicates) and must be immutable (cannot be changed).  However, a set itself is mutable. We can add or remove items from it.  Sets can also be used to perform mathematical set operations like union, intersection, symmetric difference, etc. Creating Python Sets A set is created by placing all the items (elements) inside curly braces {}, separated by comma, or by using the built-in set() function.  It can have any number of items and they may be of different types (integer, float, tuple, string etc.). But a set cannot have mutable elements like [lists](https://www.programiz.com/python-programming/list), sets or [dictionaries](https://www.programiz.com/python-programming/dictionary) as its elements.  # Different types of sets in Python  # set of integers  my\_set = {1, 2, 3}  print(my\_set)  # set of mixed datatypes  my\_set = {1.0, "Hello", (1, 2, 3)}  print(my\_set)  Run Code  **Output**  {1, 2, 3}  {1.0, (1, 2, 3), 'Hello'}  Try the following examples as well.  # set cannot have duplicates  # Output: {1, 2, 3, 4}  my\_set = {1, 2, 3, 4, 3, 2}  print(my\_set)  # we can make set from a list  # Output: {1, 2, 3}  my\_set = set([1, 2, 3, 2])  print(my\_set)  # set cannot have mutable items  # here [3, 4] is a mutable list  # this will cause an error.  my\_set = {1, 2, [3, 4]}  Run Code  **Output**  {1, 2, 3, 4}  {1, 2, 3}  Traceback (most recent call last):  File "<string>", line 15, in <module>  my\_set = {1, 2, [3, 4]}  TypeError: unhashable type: 'list'  Creating an empty set is a bit tricky.  Empty curly braces {} will make an empty dictionary in Python. To make a set without any elements, we use the set() function without any argument.  # Distinguish set and dictionary while creating empty set  # initialize a with {}  a = {}  # check data type of a  print(type(a))  # initialize a with set()  a = set()  # check data type of a  print(type(a))  Run Code  **Output**  <class 'dict'>  <class 'set'> Modifying a set in Python Sets are mutable. However, since they are unordered, indexing has no meaning.  We cannot access or change an element of a set using indexing or slicing. Set data type does not support it.  We can add a single element using the add() method, and multiple elements using the update() method. The update() method can take [tuples](https://www.programiz.com/python-programming/tuple), lists, [strings](https://www.programiz.com/python-programming/string) or other sets as its argument. In all cases, duplicates are avoided.  # initialize my\_set  my\_set = {1, 3}  print(my\_set)  # if you uncomment line 9,  # you will get an error  # TypeError: 'set' object does not support indexing  # my\_set[0]  # add an element  # Output: {1, 2, 3}  my\_set.add(2)  print(my\_set)  # add multiple elements  # Output: {1, 2, 3, 4}  my\_set.update([2, 3, 4])  print(my\_set)  # add list and set  # Output: {1, 2, 3, 4, 5, 6, 8}  my\_set.update([4, 5], {1, 6, 8})  print(my\_set)  Run Code  **Output**  {1, 3}  {1, 2, 3}  {1, 2, 3, 4}  {1, 2, 3, 4, 5, 6, 8} Removing elements from a set A particular item can be removed from a set using the methods discard() and remove().  The only difference between the two is that the discard() function leaves a set unchanged if the element is not present in the set. On the other hand, the remove() function will raise an error in such a condition (if element is not present in the set).  The following example will illustrate this.  # Difference between discard() and remove()  # initialize my\_set  my\_set = {1, 3, 4, 5, 6}  print(my\_set)  # discard an element  # Output: {1, 3, 5, 6}  my\_set.discard(4)  print(my\_set)  # remove an element  # Output: {1, 3, 5}  my\_set.remove(6)  print(my\_set)  # discard an element  # not present in my\_set  # Output: {1, 3, 5}  my\_set.discard(2)  print(my\_set)  # remove an element  # not present in my\_set  # you will get an error.  # Output: KeyError | | | |