**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **29th may 2020** | **Name:** | **Rashmitha** |
| **Course:** | **Logic design** | **USN:** | **4AL17EC077** |
| **Topic:** | **Analysis of clocked sequential circuits & digital clock design** | **Semester & Section:** | **6th sem ‘B’ sec** |
| **Github Repository:** | **Rashmitha** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session**  C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot (182).png  C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot (184).png  **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 flipflops 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).  **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.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.  **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  **DIGITAL CLOCK DESIGN**  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 outputs connect to the BCD to Seven Segment circuit 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.SECONDS blockThe 74LS93 is used to implement the divide by 10 and divide by 6 circuits. The 74LS93 is a high-speed 4-bit ripple type counters partitioned into two sections. The counter has a divide-by-two section and divide-by-eight section which are triggered by a HIGH-to-LOW transition on the clock inputs. Divide by 10 Counter In order to use all 4 bits of the counter, Q0 must be connected to CP1. Q0 is LSB and Q3 is MSB. |
|  |

**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **29th may 2020** | **Name:** | **Rashmitha** |
| **Course:** | **Python** | **USN:** | **4AL17EC077** |
| **Topic:** | **object oriented programming** | **Semester & Section:** | **6th sem ‘B’ sec** |
| **Github Repository:** | **Rashmitha** |  |  |

|  |
| --- |
| **AFTERNOON SESSION DETAILS** |
| **Image of session**  C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot (185).png  C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot (186).png  **Object-oriented programming (OOP):**  Itis a programming paradigm based on the concept of "objects", which can contain data, in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods). A feature of objects is an object's procedures that can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OOP, computer programs are designed by making them out of objects that interact with one another.OOP languages are diverse, but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.  **The Basic OOP Concepts:**  If you are new to object-oriented programming languages, you will need to know a few basics before you can get started with code. The following Webopedia definitions will help you better understand object-oriented programming:  Abstraction: The process of picking out (abstracting) common features of objects and procedures. Class: A category of objects. The class defines all the common properties of the different objects that belong to it. Encapsulation: The process of combining elements to create a new entity. A procedure is a type of encapsulation because it combines a series of computer instructions. Information hiding: The process of hiding details of an object or function. Information hiding is a powerful programming technique because it reduces complexity. Inheritance: a feature that represents the "is a" relationship between different classes. Interface: the languages and codes that the applications use to communicate with each other and with the hardware.  OOPL - Object Oriented Programming Languages An object-oriented programming language (OOPL) is a high-level programming language based on the object-oriented model. To perform object-oriented programming, one needs an object-oriented programming language. Many modern programming languages are object-oriented, however some older programming languages, such as Pascal, do offer object-oriented versions. Examples of objectoriented programming languages include Java, C++ and Smalltalk.  Object oriented programming (OOP) is a programming structure where programs are organized around objects as opposed to action and logic. This is essentially a design philosophy that uses a different set of programming languages such as C#. Understanding OOP concepts can help make decisions about how you should design an application and what language to use.  Everything in OOP is placed together as self-sustainable “objects.” An object is a combination of variables, functions, and data that performs a set of related activities. When the object performs those activities, it defines the object’s behavior. In addition, an object is an instance of a class. Furthermore, C# offers full support for OOP including inheritance, encapsulation, abstraction, and polymorphism:  Encapsulation is when a group of related methods, properties, and other members are treated as a single object. Inheritance is the ability to receive (“inherit”) methods and properties from an existing class. Polymorphism is when each class implements the same methods in varying ways, but you can still have several classes that can be utilized interchangeably. Abstraction is the process by which a developer hides everything other than the relevant data about an object in order to simplify and increase efficiency. We’ll discuss each of these concepts in more detail in this post. |
|  |