**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **13/06/2020** | **Name:** | **Varshini MN** |
| **Course:** | **VLSI design** | **USN:** | **4AL16EC089** |
| **Topic:** | **Digital VLSI Design Virtual lab** | **Semester & Section:** | **8th B** |
| **Github Repository:** | **varshinimn-test** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report**  A multiplexer or mux is a combinational circuits that selects several analog or digital input signals and forwards the selected input into a single output line. A multiplexer of 2n inputs has n selected lines, are used to select which input line to send to the output.    Below Figure shows how a 4:1 MUX can be constructed out of two 2:1 MUXs.    **LOGIC GATES:**  **Static logic** is a design methodology in integrated circuit design where there is at all times some mechanism to drive the output either high or low. For example, in many of the popular logic families, such as TTL and traditional CMOS, there is always a low-impedance path between the output and either the supply voltage or the ground. The most widely used logic style is static CMOS. A static CMOS gate is a combination of two networks, called the pull-up network (PUN) and the pull-down network (PDN). The function of the PUN is to provide a connection between the output and VDD anytime the output of the logic gate is meant to be 1 (based on the inputs). Similarly, the function of the PDN is to connect the output to VSS when the output of the logic gate is meant to be 0 (based on the inputs). The PUN and PDN networks are constructed in a mutually exclusive fashion such that, one and only one of these networks is conducting in the steady state.    **MOSFET:**  The metal–oxide–semiconductor field-effect transistor (MOSFET) is a transistor used for amplifying or switching electronic signals. In MOSFETs, a voltage on the oxide-insulated gate electrode can induce a conducting channel between the two other contacts called source and drain. The channel can be of n-type or p-type, and is accordingly called an nMOSFET or a pMOSFET. Figure 1 shows the schematic diagram of the structure of an nMOS device before and after channel formation.      Figure 2 shows symbols commonly used for MOSFETs where the bulk terminal is either labeled (B) or implied (not drawn).    Fig. (2): Circuit symbols for nMOS and pMOS respectively |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **13/06/2020** | **Name:** | **Varshini MN** | |
| **Course:** | **Java** | **USN:** | **4AL16EC089** | |
| **Topic:** | **The Equals Method, Inner classes..** | **Semester & Section:** | **8th B** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Report**  **Serialization**  Scan each element in a string, calculate its length and append it with a string and a element separator or deliminator (that deliminator should not be present in the string). We append the length of the string so that we know the length of each element.  **Deserialized Function**  Find the position of the deliminator, then from the position + 1 to length of word we store it in an array as a single element.  // CPP program to serialize and  // deserialize the array of string  #include<iostream>  using namespace std;  // Function to serialized the array of string  string serialize(string str[], int ln)  {  string temp = "";  for (int i=0; i<ln; i++)  {  int ln = str[i].length();  temp.push\_back('0' + ln);  temp = temp + "~" + str[i];  }  return temp;  }  // Function to deserialize the string  void deserialized(string str, string deserialize[], int ln)  {  int len, pos=0;  string temp = "";  int i = 0;  while(pos>-1)  {  pos = str.find("~", pos+1);  if(pos>0)  {  len = str[pos-1] - 48;  temp.append(str, pos+1, len);  deserialize[i++] = temp;  temp = "";  }  }  }  // Driver function  int main()  {  string str[] = {"geeks", "are", "awesome"};  int ln = sizeof(str)/sizeof(str[0]);  string serializedstr = serialize(str, ln);  cout<< "Serialized String : " <<serializedstr<<endl;  string deserialize[ln];  deserialized(serializedstr,deserialize,ln);  cout<< "Deserialized String : ";  for(int i=0; i<ln; i++)  cout<< deserialize[i] << " ";  return 0;  } **Equals method in Java** class Complex {  private double re, im;    public Complex(double re, double im) {  this.re = re;  this.im = im;  }  }  // Driver class to test the Complex class  public class Main {  public static void main(String[] args) {  Complex c1 = new Complex(10, 15);  Complex c2 = new Complex(10, 15);  if (c1 == c2) {  System.out.println("Equal ");  } else {  System.out.println("Not Equal ");  }  }  } | | | |