**ASSESSMENT 20**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | 09-06-2020 | **Name:** | Sheela Golasangi |
| **Course:** | VLSI | **USN:** | 4AL16EC068 |
| **Topic:** | 1. MOSFET - Enhancement Type MOSFET Explained (Construction, Working and Characteristics Explained) 2. GATE 2009 and 20121 ECE operating region and output voltage of CMOS inverter given 3. MOSFET vth based problems 4. MOSFET problems and solutions 5. TRICK to implement 4:1 mux using TRANSMISSION GATE & PASS TRANSISTOR LOGIC 6. MOSFET Drain current - graph , formulae & sums (cutoff,linear& saturation) 7. Realization of logic function using Multiplexer | **Semester & Section:** | VIII  ‘B’ |
| **Github Repository:** | Sheela-Course |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| MOSFET In case of JFET, the gate must be reverse biased for proper operation of the device i.e. it can only have negative gate operation for n-channel and positive gate operation for p-channel. That means we can only decrease the width of the channel from its zero-bias size. This type of operation is known as depletion-mode operation. Therefore, a JFET can only be operated in the depletion mode.  However, there is a field effect transistor that can be operated to enhance the width of the channel i.e. it can have enhancement-mode operation. Such a FET is called MOSFET. Types of MOSFETs There are two basic types of MOSFETs such as:   1. Depletion-type MOSFET or D-MOSFET: The D-MOSFET can be operated in both depletion mode and the enhancement mode. For this reason it is also called depletion/enhancement MOSFET. 2. Enhancement-type MOSFET or E-MOSFET: The E-MOSFET can be operated only in enhancement mode.  ****Depletion mode:**** Since gate is negative, it means electrons are on the gate  These electrons repel the free electrons in the n-channel, leaving  a layer of positive ions in a part of the channel as shown in fig.5 (ii).In other words, the n-channel is depleted of some of its free electrons.  Therefore, lesser number of free electrons are available for current conduction through the n-channel. This is same as increasing the channel resistance.  The greater the negative voltage on the gate, the lesser is the current from source to drain.  Thus by changing the negative voltage on the gate, we can vary the resistance of the n-channel and hence the current from source to drain. As the action with negative gate depends upon depleting the channel of free electrons, the negative-gate operation is called depletion mode.2. Enhancement mode:Again the gate acts like a capacitor. Since the gate is positive, it induces negative charges in the n-channelThese negative charges are the free electrons drawn into the channel.Because these free electrons are added to those already in the channel, the total number of free electrons in the channel is increased.Thus a positive gate voltage enhances or increases the conductivity of the channel.The greater the positive voltage on the gate, greater the conduction from source to drain.Thus by changing the positive voltage on the gate, we can change the conductivity of the channel. Because the action with a positive gate depends upon enhancing the conductivity of the channel, the positive gate operation is called enhancement mode.  **GATE PROBLUMNS AD SPLUTIONS**      **REALIZATION OF LOGIC FUNCTION USING MULTIPLEXER**    **Multiplexers in Digital Logic**  It is combinational circuits which have many data inputs and single output depending on control or select inputs.​ For N input lines, log n (base2) selection lines, or we can say that for 2n input lines, n selection lines are required. Multiplexers are also known as “Data n selector, parallel to serial convertor, many to one circuit, universal logic circuit”. Multiplexers are mainly used to increase amount of the data that can be sent over the network within certain amount of time and bandwidth. Now the implementation of 4:1 Multiplexer using truth table and gates. |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | 09-06-2020 | **Name:** | Sheela Golasangi |
| **Course:** | Java Tutorial for Complete Beginners | **USN:** | 4AL16EC068 |
| **Topic:** | Programming core java   1. A Hello World Program 2. Using Variables 3. Strings: Working With Text 4. While Loops 5. For Loops 6. "If" 7. Getting User Input 8. Do ... While 9. Switch 10. Arrays | **Semester & Section:** | VIII  ‘B’ |
| **Github Repository:** | Sheela-Course |  |  |
| **AFTERNOON SESSION DETAILS** | | | |
| **PROGRAMMING CORE JAVA**  **A Hello World Program**    A "Hello, World!" is a simple program that outputs Hello, World! on the screen. Since it's a very simple program, it's often used to introduce a new programming language to a newbie.  Let's explore how Java "Hello, World!" program works.  If you want to run this program on your computer, make sure that Java is properly installed. Also, you need an IDE (or a text editor) to write and edit Java code.  Java "Hello, World!" Program    class HelloWorld {  public static void main(String[] args) {  System.out.println("Hello, World!");  }  }  If you have copied the exact code, you need to save the file name as HelloWorld.java. It's because the name of the class and filename should match in Java.  **Using Variables** Java Variables A variable is a container which holds the value while the [Java program](https://www.javatpoint.com/simple-program-of-java) is executed. A variable is assigned with a data type.  Variable is a name of memory location. There are three types of variables in java: local, instance and static.  There are two types of [data types in Java](https://www.javatpoint.com/java-data-types): primitive and non-primitive. Variable **Variable** is name of reserved area allocated in memory. In other words, it is a name of memory location. It is a combination of "vary + able" that means its value can be changed. Types of Variables There are three types of variables in [Java](https://www.javatpoint.com/java-tutorial):   * local variable * instance variable * static variable  1) Local Variable A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.  A local variable cannot be defined with "static" keyword. 2) Instance Variable A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as [static](https://www.javatpoint.com/static-keyword-in-java).  It is called instance variable because its value is instance specific and is not shared among instances. 3) Static variable A variable which is declared as static is called static variable. It cannot be local. You can create a single copy of static variable and share among all the instances of the class. Memory allocation for static variable happens only once when the class is loaded in the memory. Example to understand the types of variables in java  1. **class** A{ 2. **int** data=50;//instance variable 3. **static** **int** m=100;//static variable 4. **void** method(){ 5. **int** n=90;//local variable 6. } 7. }//end of class   **Strings: Working With Text, While Loops, For Loops, "If", Getting User Input, Do ... While**  **Strings:**    we learned how to use System.out.printf to display formatted output. Sometimes programs need to create strings that are formatted a certain way, but not display them immediately, or ever. For example, the following method returns a time string in 12-hour format:  public static String timeString(int hour, int minute) {  String ampm;  if (hour < 12) {  ampm = "AM";  if (hour == 0) {  hour = 12; // midnight  }  } else {  ampm = "PM";  hour = hour - 12;  }  return String.format("%02d:%02d %s", hour, minute, ampm);  }  String.format takes the same arguments as System.out.printf: a format specifier followed by a sequence of values. The main difference is that System.out.printf displays the result on the screen. String.format creates a new string, but does not display anything.  In this example, the format specifier \%02d means “two digit integer padded with zeros”, so time String (19, 5) returns the string "07:05 PM". As an exercise, try writing two nested for loops (in main) that invoke timeString and display all possible times over a 24-hour period.  At some point today, skim through the documentation for String. Knowing what other methods are there will help you avoid reinventing the wheel. The easiest way to find documentation for Java classes is to do a web search for “Java” and the name of the class  **While& Do while Loop:**  **While**  In this tutorial, we will learn how to use while and do while loop in Java with the help of examples and we will also learn about the working of Loop in computer programming In computer programming, loops are used to repeat a specific block of code until a certain condition is met (test expression is false). For example,  Imagine we need to print a sentence 50 times on your screen. Well, we can do it by using the print statement 50 times (without using loops). How about you need to print a sentence one million times? You need to use loops. With loops, we can simply write the print statement one time and run it for any number of times.  It's just a simple example showing the importance of loop in computer programming. There are 3 types of loops in Java: for loop, while loop, and do-while loop.  Java while Loop  The syntax of while loop in Java is:  while (testExpression) {  // codes inside the body of while loop  }  **Do While**  The do...while loop is similar to while loop with one key difference. The body of do...while loop is executed for once before the test expression is checked.  Here is the syntax of the do...while loop.  do {  // codes inside body of do while loop  } while (testExpression);  **For loop**  Loops are used to execute a set of statements repeatedly until a particular condition is satisfied. In Java we have three types of basic loops: for, while and do-while. In this tutorial we will learn how to use “for loop” in Java.  Syntax of for loop:  for(initialization; condition ; increment/decrement)  {  statement(s);  }  **if** Java If-else Statement The [Java](https://www.javatpoint.com/java-tutorial) if statement is used to test the condition. It checks [boolean](https://www.javatpoint.com/boolean-keyword-in-java) condition: trueorfalse*.* There are various types of if statement in Java.   * if statement * if-else statement * if-else-if ladder * nested if statement  Java if Statement The Java if statement tests the condition. It executes the if block if condition is true.   1. **Syntax:**if(condition){ 2. //code to be executed 3. }   Example:   1. //Java Program to demonstate the use of if statement. 2. public class IfExample { 3. public static void main(String[] args) { 4. //defining an 'age' variable 5. int age=20; 6. //checking the age 7. if(age>18){ 8. System.out.print("Age is greater than 18"); 9. } 10. } 11. }   **Switch statement**   Java Switch Statement The Java switch statement executes one statement from multiple conditions. It is like [if-else-if](https://www.javatpoint.com/java-if-else) ladder statement. The switch statement works with byte, short, int, long, enum types, String and some wrapper types like Byte, Short, Int, and Long. Since Java 7, you can use [strings](https://www.javatpoint.com/java-string) in the switch statement.  In other words, the switch statement tests the equality of a variable against multiple values. Points to Remember  * There can be one or N number of case values for a switch expression. * The case value must be of switch expression type only. The case value must be literal or constant. It doesn't allow [variables](https://www.javatpoint.com/java-variables). * The case values must be unique. In case of duplicate value, it renders compile-time error. * The Java switch expression must be of byte, short, int, long (with its Wrapper type), [enums](https://www.javatpoint.com/java-switch) and string. * Each case statement can have a break statement which is optional. When control reaches to the [break statement](https://www.javatpoint.com/java-break), it jumps the control after the switch expression. If a break statement is not found, it executes the next case. * The case value can have a default label which is optional.   **Syntax:**   1. switch(expression){ 2. case value1: 3. //code to be executed; 4. break;  //optional 5. case value2: 6. //code to be executed; 7. break;  //optional 8. ...... 10. default: 11. code to be executed if all cases are not matched; 12. }   **Arrays**    An array is a group of like-typed variables that are referred to by a common name.Arrays in Java work differently than they do in C/C++. Following are some important point about Java arrays.   * In Java all arrays are dynamically allocated.(discussed below) * Since arrays are objects in Java, we can find their length using member length. This is different from C/C++ where we find length using sizeof. * A Java array variable can also be declared like other variables with [] after the data type. * The variables in the array are ordered and each has an index beginning from 0. * Java array can be also be used as a static field, a local variable or a method parameter. * The **size** of an array must be specified by an int value and not long or short. * The direct superclass of an array type is [Object](https://www.geeksforgeeks.org/object-class-in-java/). * Every array type implements the interfaces [Cloneable](https://www.geeksforgeeks.org/marker-interface-java/) and [java.io.Serializable](https://www.geeksforgeeks.org/serialization-in-java/).   Array can contain primitives (int, char, etc) as well as object (or non-primitives) references of a class depending on the definition of array. In case of primitive’s data types, the actual values are stored in contiguous memory locations. In case of objects of a class, [the actual objects are stored in heap segment](https://www.geeksforgeeks.org/g-fact-46/). | | | |