**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **11/06/2020** | **Name:** | **Navya R** |
| **Course:** | **VLSI design** | **USN:** | **4AL16EC041** |
| **Topic:** | **CMOS Inverter Basics** | **Semester & Section:** | **8 A** |
| **Github Repository:** | **nayana\_online** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report**  **The Static CMOS Inverter –**  Its operation is readily understood with the aid of the simple switch model of the MOS transistor, : the transistor is nothing more than a switch with an infinite off resistance (for |VGS| < |VT |), and a finite on-resistance (for |VGS| > |VT |). This leads to the following interpretation of the inverter. When Vin is high and equal to VDD, the NMOS transistor is on, while the PMOS is off. This yields the equivalent circuit of Figure a. A direct path exists between Vout and the ground node, resulting in a steady-state value of 0 V. On the other hand, when the input voltage is low (0 V), NMOS and PMOS transistors are off and on, respectively. The equivalent circuit of Figure b shows that a path exists between VDD and Vout, yielding a high output voltage. The gate clearly functions as an inverter.  A number of other important properties of static CMOS can be derived from this switch level view: • The high and low output levels equal VDD and GND, respectively; in other words, the voltage swing is equal to the supply voltage. This results in high noise margins. • The logic levels are not dependent upon the relative device sizes, so that the transistors can be minimum size. Gates with this property are called ratio less. This is in contrast with ratioed logic, where logic levels are determined by the relative dimensions of the composing transistors. • In steady state, there always exists a path with finite resistance between the output and either VDD or GND. A well-designed CMOS inverter, therefore, has a low output impedance, which makes it less sensitive to noise and disturbances. Typical values of the output resistance are in kW range. • The input resistance of the CMOS inverter is extremely high, as the gate of an MOS transistor is a virtually perfect insulator and draws no dc input current.  Since the input node of the inverter only connects to transistor gates, the steady-state input current is nearly zero. A single inverter can theoretically drive an infinite number of gates (or have an infinite fan-out) and still be functionally operational; however, increasing the fan-out also increases the propagation delay, as will become clear below. So, although fan-out does not have any effect on the steady-state behavior, it degrades the transient response.    The nature and the form of the voltage-transfer characteristic (VTC) can be graphically deduced by superimposing the current characteristics of the NMOS and the PMOS devices. Such a graphical construction is traditionally called a load-line plot. It requires that the I-V curves of the NMOS and PMOS devices are transformed onto a common coordinate set. We have selected the input voltage Vin, the output voltage Vout and the NMOS drain current IDN as the variables of choice. The PMOS I-V relations can be translated into this variable space by the following relations (the subscripts n and p denote the NMOS and PMOS devices, respectively): (5.1) The load-line curves of the PMOS device are obtained by a mirroring around the xaxis and a horizontal shift over VDD. This procedure is outlined in Figure 5.3, where the subsequent steps to adjust the original PMOS I-V curves to the common coordinate set Vin, Vout and IDn are illustrated. IDSp I = – DSn VGSn Vin VGSp = ; = Vin – VDD VDSn Vout VDSp = ; = Vout – VDD |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **11/6/2020** | **Name:** | **Nayanashree K S** | |
| **Course:** | **Java** | **USN:** | **4al16ec042** | |
| **Topic:** | **The to String Method**  **Inheritance**  **Packages…….** | **Semester & Section:** | **8 A** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **REPORT** **Java - toString() Method** The method is used to get a String object representing the value of the Number Object.  If the method takes a primitive data type as an argument, then the String object representing the primitive data type value is returned.  If the method takes two arguments, then a String representation of the first argument in the radix specified by the second argument will be returned. Syntax Following are all the variants of this method −  String toString()  static String toString(int i) Parameters Here is the detail of parameters −   * **i** − An int for which string representation would be returned.  Return Value  * **toString()** − This returns a String object representing the value of **this** Integer. * **toString(int i)** − This returns a String object representing the specified integer.  Example public class Test {  public static void main(String args[]) {  Integer x = 5;  System.out.println(x.toString());  System.out.println(Integer.toString(12));  }  }  This will produce the following result − Output 5  12 **Inheritance in Java**  1. [Inheritance](https://www.javatpoint.com/inheritance-in-java) 2. [Types of Inheritance](https://www.javatpoint.com/inheritance-in-java#inheritancetypes) 3. [Why multiple inheritance is not possible in Java in case of class?](https://www.javatpoint.com/inheritance-in-java#inheritancenotmultiple)   **Inheritance in Java** is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of [OOPs](https://www.javatpoint.com/java-oops-concepts) (Object Oriented programming system).  The idea behind inheritance in Java is that you can create new [classes](https://www.javatpoint.com/object-and-class-in-java) that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.  Inheritance represents the **IS-A relationship** which is also known as a parent-child relationship. **The syntax of Java Inheritance**  1. **class** Subclass-name **extends** Superclass-name 2. { 3. //methods and fields 4. }   The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.  In the terminology of Java, a class which is inherited is called a parent or superclass, and the new class is called child or subclass. | | | |