**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **13-June-2020** | **Name:** | **Raziya Banu** |
| **Course:** | **VLSI** | **USN:** | **4AL16EC058** |
| **Topic:** | **CMOS Inverter Basics** | **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 CMOS Inverter Basics . **CMOS Working Principle and Applications** The term CMOS stands for “Complementary Metal Oxide Semiconductor”. CMOS technology is one of the most popular technology in the computer chip design industry and broadly used today to form [integrated circuits](https://www.elprocus.com/how-integrated-circuits-work-physically/) in numerous and varied applications. Today’s computer memories, CPUs and cell phones make use of this technology due to several key advantages. This technology makes use of both P channel and N channel semiconductor devices.  One of the most popular MOSFET technologies available today is the Complementary MOS or CMOS technology. This is the dominant semiconductor technology for microprocessors, microcontroller chips, memories like RAM, ROM, [EEPROM and application](https://www.elprocus.com/eeprom-features-applicaitons-circuit-diagram/) specific integrated circuits (ASICs). CMOS (Complementary Metal Oxide Semiconductor) The main [advantage of CMOS over NMOS](https://www.elprocus.com/difference-between-nmos-cmos-technology/) and BIPOLAR technology is the much smaller power dissipation. Unlike NMOS or BIPOLAR circuits, a Complementary MOS circuit has almost no static power dissipation. Power is only dissipated in case the circuit actually switches. This allows integrating more CMOS gates on an IC than in NMOS or[bipolar technology](https://www.elprocus.com/bipolar-junction-transistors-working-principle-and-applications/), resulting in much better performance. Complementary Metal Oxide Semiconductor transistor consists of P-channel MOS (PMOS) and N-channel MOS (NMOS). Please refer the link to know more about [the fabrication process of CMOS transistor](https://www.elprocus.com/the-fabrication-process-of-cmos-transistor/).  CMOS Transistor  **CMOS Transistor** NMOS NMOS is built on a p-type substrate with n-type source and drain diffused on it. In NMOS, the majority carriers are electrons. When a high voltage is applied to the gate, the NMOS will conduct. Similarly, when a low voltage is applied to the gate, NMOS will not conduct. NMOS are considered to be faster than PMOS, since the carriers in NMOS, which are electrons, travel twice as fast as the holes.  NMOS Transistor  NMOS Transistor PMOS P- channel MOSFET consists P-type Source and Drain diffused on an N-type substrate. Majority carriers are holes. When a high voltage is applied to the gate, the PMOS will not conduct. When a low voltage is applied to the gate, the PMOS will conduct. The PMOS devices are more immune to noise than NMOS devices.  PMOS Transistor  PMOS Transistor CMOS Working Principle In CMOS technology, both N-type and P-type transistors are used to design logic functions. The same signal which turns ON a transistor of one type is used to turn OFF a transistor of the other type. This characteristic allows the design of logic devices using only simple switches, without the need for a pull-up resistor.  In CMOS [logic gates](https://www.elprocus.com/basic-logic-gates-with-truth-tables/) a collection of n-type MOSFETs is arranged in a pull-down network between the output and the low voltage power supply rail (Vss or quite often ground). Instead of the load resistor of NMOS logic gates, CMOS logic gates have a collection of p-type MOSFETs in a pull-up network between the output and the higher-voltage rail (often named Vdd).  Thus, if both a p-type and n-type transistor have their gates connected to the same input, the p-type MOSFET will be ON when the n-type MOSFET is OFF, and vice-versa. The networks are arranged such that one is ON and the other OFF for any input pattern as shown in the figure below.  CMOS Logic Gate using Pull-Up and Pull-Down Networks  CMOS Logic Gate using Pull-Up and Pull-Down Networks  CMOS offers relatively high speed, low power dissipation, high noise margins in both states, and will operate over a wide range of source and input voltages (provided the source voltage is fixed). Furthermore, for the better understanding of the Complementary Metal Oxide Semiconductor working principle, we need to discuss in brief about CMOS logic gates as explained below. **CMOS Inverter** The inverter circuit as shown in the figure below. It consists of [PMOS and NMOS FET](https://www.elprocus.com/mosfet-as-a-switch-circuit-diagram-free-circuits/). The input A serves as the gate voltage for both transistors.  CMOS Inverter  CMOS Inverter  The NMOS transistor has an input from Vss (ground) and PMOS transistor has an input from Vdd. The terminal Y is output. When a high voltage (~ Vdd) is given at input terminal (A) of the inverter, the PMOS becomes open circuit and NMOS switched OFF so the output will be pulled down to Vss.  When a low-level voltage (<Vdd, ~0v) applied to the inverter, the NMOS switched OFF and PMOS switched ON. So the output becomes Vdd or the circuit is pulled up to Vdd.   |  |  |  |  | | --- | --- | --- | --- | | **INPUT** | **LOGIC INPUT** | **OUTPUT** | **LOGIC OUTPUT** | | 0 v | 0 | Vdd | 1 | | Vdd | 1 | 0 v | 0 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **13-June-2020** | **Name:** | **Raziya Banu** | |
| **Course:** | **Udemy** | **USN:** | **4AL16EC058** | |
| **Topic:** | **Programming core Java** | **Semester & Section:** | **8th sem & ‘B’ section** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| **Arrays of Strings** A tutorial on String arrays in Java, plus another way to iterate through an array, and more stuff on references vs. values.  **public** **class** **App** **{**  **public** **static** **void** **main(**String**[]** args**)** **{**    *// Declare array of (references to) strings.*  String**[]** words **=** **new** String**[**3**];**    *// Set the array elements (point the references*  *// at strings)*  words**[**0**]** **=** "Hello"**;**  words**[**1**]** **=** "to"**;**  words**[**2**]** **=** "you"**;**    *// Access an array element and print it.*  System**.**out**.**println**(**words**[**2**]);**    *// Simultaneously declare and initialize an array of strings*  String**[]** fruits **=** **{**"apple"**,** "banana"**,** "pear"**,** "kiwi"**};**    *// Iterate through an array*  **for(**String fruit: fruits**)** **{**  System**.**out**.**println**(**fruit**);**  **}**    *// "Default" value for an integer*  **int** value **=** 0**;**    *// Default value for a reference is "null"*  String text **=** **null;**    System**.**out**.**println**(**text**);**    *// Declare an array of strings*  String**[]** texts **=** **new** String**[**2**];**    *// The references to strings in the array*  *// are initialized to null.*  System**.**out**.**println**(**texts**[**0**]);**    *// ... But of course we can set them to actual strings.*  texts**[**0**]** **=** "one"**;**  **}**  **}**    you  apple  banana  pear  kiwi  null  null  **Multi-dimensional Arrays** [**Java for Complete Beginners**](https://www.caveofprogramming.com/categories/java-video/index.html) How to work with multi-dimensional arrays in Java and info on how multi-dimensional arrays actually work, plus some tips on how to remember which index is which.  **public** **class** **App** **{**  **public** **static** **void** **main(**String**[]** args**)** **{**    *// 1D array*  **int[]** values **=** **{**3**,** 5**,** 2343**};**    *// Only need 1 index to access values.*  System**.**out**.**println**(**values**[**2**]);**    *// 2D array (grid or table)*  **int[][]** grid **=** **{**  **{**3**,** 5**,** 2343**},**  **{**2**,** 4**},**  **{**1**,** 2**,** 3**,** 4**}**  **};**    *// Need 2 indices to access values*  System**.**out**.**println**(**grid**[**1**][**1**]);**  System**.**out**.**println**(**grid**[**0**][**2**]);**    *// Can also create without initializing.*  String**[][]** texts **=** **new** String**[**2**][**3**];**    texts**[**0**][**1**]** **=** "Hello there"**;**    System**.**out**.**println**(**texts**[**0**][**1**]);**    *// How to iterate through 2D arrays.*  *// first iterate through rows, then for each row*  *// go through the columns.*  **for(int** row**=**0**;** row **<** grid**.**length**;** row**++)** **{**  **for(int** col**=**0**;** col **<** grid**[**row**].**length**;** col**++)** **{**  System**.**out**.**print**(**grid**[**row**][**col**]** **+** "\t"**);**  **}**    System**.**out**.**println**();**  **}**    *// The last array index is optional.*  String**[][]** words **=** **new** String**[**2**][];**    *// Each sub-array is null.*  System**.**out**.**println**(**words**[**0**]);**    *// We can create the subarrays 'manually'.*  words**[**0**]** **=** **new** String**[**3**];**  words**[**0**][**1**]** **=** "hi there"**;**    System**.**out**.**println**(**words**[**0**][**1**]);**  **}**  **}**    2343  4  2343  Hello there  3 5 2343  2 4  1 2 3 4  null  hi there | | | |