**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **12-06-2020** | **Name:** | **Anand kumar k** |
| **Course:** |  | **USN:** | **4al16ec002** |
| **Topic:** | **VLSI** | **Semester & Section:** | **8thsem ‘A’ sec** |
| **Github Repository:** | **Anand-courses** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session**      The inverter is truly the nucleus of all digital designs. Once its operation and properties are clearly understood, designing more intricate structures such as NAND gates, adders, multipliers, and microprocessors is greatly simplified. The electrical behavior of these complex circuits can be almost completely derived by extrapolating the results obtained for inverters. The analysis of inverters can be extended to explain the behavior of more complex gates such as NAND, NOR, or XOR, which in turn form the building blocks for modules such as multipliers and processors  cost, expressed by the complexity and area  • integrity and robustness, expressed by the static (or steady-state) behavior  • performance, determined by the dynamic (or transient) response  • energy efficiency, set by the energy and power consumption From this analysis arises a model of the gate that will help us to identify the parameters of the gate and to choose their values so that the resulting design meets desired specifications. While each of these parameters can be easily quantified for a given technology, we also discuss how they are affected by scaling of the technology.  A number of other important properties of static CMOS can be derived from this switchlevel 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 ratioless. 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.  • In the following sections, we will learn that reducing the supply voltage indiscriminately has a positive impact on the energy dissipation, but is absolutely detrimental to the performance on the gate.  • The dc-characteristic becomes increasingly sensitive to variations in the device parameters such as the transistor threshold, once supply voltages and intrinsic voltages become comparable.  • Scaling the supply voltage means reducing the signal swing. While this typically helps to reduce the internal noise in the system (such as caused by crosstalk), it makes the design more sensitive to external noise sources that do not scale.  Manual analysis of MOS circuits where each capacitor is considered individually is virtually impossible and is exacerbated by the many nonlinear capacitances in the MOS transistor model. To make the analysis tractable, we assume that all capacitances are lumped together into one single capacitor CL , located between Vout and GND. Be aware that this is a considerable simplification of the actual situation, even in the case of a simple inverter.  The capacitance between drain and bulk is due to the reverse-biased pn-junction. Such a capacitor is, unfortunately, quite nonlinear and depends heavily on the applied voltage. We argued in Chapter 3 that the best approach towards simplifying the analysis is to replace the nonlinear capacitor by a linear one with the same change in charge for the voltage range of interest. A multiplication factor Keq is introduced to relate the linearized capacitor to the value of the junction capacitance under zero-bias conditions. |
|  |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **12-06-2020** | **Name:** | **Anand kumar k** | |
| **Course:** |  | **USN:** | **4al16ec002** | |
| **Topic:** | **mysql** | **Semester & Section:** | **8thsem ‘A’ sec** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| Introduction to MySQL LOOP statement The LOOP statement allows you to execute one or more statements repeatedly.  Here is the basic syntax of the LOOP statement:  [begin\_label:] LOOP  statement\_list  END LOOP [end\_label]  The LOOP can have optional labels at the beginning and end of the block.  The LOOP executes the statement\_list repeatedly. The statement\_list may have one or more statements, each terminated by a semicolon (;) statement delimiter.  Typically, you terminate the loop when a condition is satisfied by using the [LEAVE](https://www.mysqltutorial.org/mysql-stored-procedure/mysql-leave/) statement.  This is the typical syntax of the LOOP statement used with LEAVE statement:  [label]: LOOP  ...  *-- terminate the loop*  IF condition THEN  LEAVE [label];  END IF;  ...  END LOOP;  The LEAVE statement immediately exits the loop. It works like the break statement in other programming languages like PHP, C/C++, and Java.  In addition to the LEAVE statement, you can use the ITERATE statement to skip the current loop iteration and start a new iteration. The ITERATE is similar to the continue statement in PHP, C/C++, and Java.  The WHILE loop is a loop statement that executes a block of code repeatedly as long as a condition is true.  Here is the basic syntax of the WHILE statement:  [begin\_label:] WHILE search\_condition DO  statement\_list  END WHILE [end\_label]  In this syntax:  First, specify a search condition after the WHILE keyword.  The WHILE checks the search\_condition at the beginning of each iteration.  If the search\_condition evaluates to TRUE, the WHILE executes the statement\_list as long as the search\_condition is TRUE.  The WHILE loop is called a pretest loop because it checks the search\_condition before the statement\_list executes.  Second, specify one or more statements that will execute between the DO and END WHILE keywords.  Third, specify optional labels for the WHILE statement at the beginning and end of the loop construct.  The following flowchart illustrates the MySQL WHILE loop statement: | | | |