**DAILY ASSESSMENT REPORT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **12/06/2020** | **Name:** | **RUSSELL** |
| **Course:** | **Vlsi** | **USN:** | **4al15EC023** |
| **Topic:** | **CMOS Inverter Basics** | **Semester & Section:** | **8th &"A" section** |
| **Github Repository:** | **Russell1005** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report–Reportcanbetypedorhandwrittenforuptotwopages.**  Complementary MOSFET (CMOS) technology is widely used to day to form circuits in numerous and varied applications.  Today’s computers CPUs and cell phones make use of CMOS due to several key advantages.  CMOS offers low power dissipation, relatively high speed, high noise margins in both states, and will operate over a wide range of source and input voltages (provided the source voltage is fixed).  Next I will attempt to explain just how this logic gate works now that you have some idea of how important CMOS is in your day-to-day life.         |  | | --- | | **CMOS Inverter Basics** |   As you can see from Figure 1, a CMOS circuit is composed of two MOSFETs.  The top FET (MP) is a PMOS type device while the bottom FET (MN) is an NMOS type.  The body effect is not present in either device since the body of each device is directly connected to the device’s source.  Both gates are connected to the input line.  The output line connects to the drains of both FETs.    Take a look at the VTC in Figure 2.  The curve represents the output voltage taken from node 3.  You can easily see that the CMOS circuit functions as an inverter by noting that when VIN is five volts, VOUT is zero, and vice versa. Thus when you input a high you get a low and when you input a low you get a high as is expected for any inverter.  You might be wondering what happens in the middle, transition area of the curve.  You might also be curious as to what modes of operation the MOSFETs are in.  We will look at these issues next.       |  |  | | --- | --- | | https://courseware.ee.calpoly.edu/~dbraun/courses/ee307/F02/02_Shelley/cmos_files/image002.gif                            Figure1:  CMOS Inverter | https://courseware.ee.calpoly.edu/~dbraun/courses/ee307/F02/02_Shelley/cmos_files/image004.jpg                Figure 2:  Basic Voltage Transfer Characteristic |          |  | | --- | | **DC Analysis** |   Figure 3 shows a more detailed VTC.  Before we begin our analysis it is important to mention three items.     * The MOSFETS must be perfectly matched for optimum operation, that is, they must have the same threshold voltage magnitude and conduction parameter. * The drain current (ID) through the NMOS device equals the drain current through the PMOS device at all times.  MOSFET gates have a high input impedance and we assume the circuit’s output sees no significant loading. * VDD equals the voltage across the PMOS plus the voltage across the NMOS by KVL.     https://courseware.ee.calpoly.edu/~dbraun/courses/ee307/F02/02_Shelley/cmos_files/image006.gif  **Figure 3:  VTC with Input Signal**    **Region I**    First we focus our attention on **region I**.  In this case when we apply an input voltage between 0 and VTN.  The PMOS device on since a low voltage is being applied to it.  The NMOS is already negative enough and has no use for more free electrons so it refuses to conduct and turns into a large resistor.  Since the NMOS device is on vacation, there is no current flow through either device.  VDD is available at the Vo terminal since no current is going through the PMOS device and thus no voltage is being dropped across it.     * The PMOS device is forward biased (VSG > -VTP) and therefore on.  This MOSFET is in the linear region (VSD<=VSG+VTP=VDD-Vo+VTP). * The NMOS device is cut off since the input voltage is below VTN (Vi=VGS<VTN). * The power dissipation is zero.     **Region II**    Here we raise the input voltage above VTN.  We find that the PMOS device remains in the linear region since it still has adequate forward bias.  The NMOS turns on and jumps immediately into saturation since it still has a relatively large VDS across it.     * The PMOS device is in the linear region (VSD<=VSG+VTP). * The NMOS device is in the saturation region (Vi=VDS>=VGS-VTN=Vo-VTN). * Current now flows through both devices.  Power dissipation is no longer zero.     The maximum allowable input voltage at the low logic state (**VIL**) occurs in this region.  VIL is the value of Vi at the point where the slope of the VTC is -1.  Put another way, VIL occurs at (dVo/dVi)=-1.    **Region III**    In the middle of this region there exists a point where Vi=Vo.  We label this point VM and identify it as the gate threshold voltage.  The voltage dropped across the NMOS device equals the voltage dropped across the PMOS device when the input voltage is VM.  For a very short time, both devices see enough forward bias voltage to drive them to saturation.     * The PMOS device is in the saturation region (VSD>=VSG+VTP=VDD-Vo+VTP). * The NMOS device is in the saturation region (VDS>=VGS-VTN=Vo-VTN). * Power dissipation reaches a peak in this region, namely at where VM=Vi=Vo.     **Region IV**    Region IV occurs between an input voltage slightly higher than VM but lower than VDD-VTP.  Now the NMOS device is conducting in the linear region, dropping a low voltage across VDS.  Since VDS is relatively low, the PMOS device must pick up the tab and drop the rest of the voltage (VDD-VDS) across its VSD junction.  This, in turn, drives the PMOS into saturation.  This region is effectively the reverse of region II.     * The PMOS device is in the saturation region (VSD>=VSG+VTP=VDD-Vo+VTP). * The NMOS device is forward biased (Vi=VGS > VTN) and therefore on.  This MOSFET is in the linear region (Vi=VDS<=VGS-VTN=Vo-VTN).     The minimum allowable input voltage at the logic high state (**VIH**) occurs in this region.  VIH occurs at the point where the slope of the VTC is –1 (dVo/dVi)=-1.    **Region V**    The NMOS wants to conduct but its drain current is severely limited due to the PMOS device only letting through a tiny leakage current.  The PMOS is out to lunch since it is seeing a positive drive but it is already positive enough and has no use for more.  This drain current let through by the PMOS is too small to matter in most practical cases so we let ID=0.  With this information we can conclude that VDS=Vo=0 V for the NMOS since no current is going through the device.  We have, in effect, sent in VDD and found the inverter’s output to be zero volts.  **For CMOS inverters, VOH=VDD.**  VOL is defined to be the output voltage of the inverter at an input voltage of VOH.  We have just proven that **VOL=0**.     * The PMOS device is cut off when the input is at VDD (VSG=0 V). * The NMOS device is forward biased (Vi=VGS > VTN) and therefore on.  This MOSFET is in the linear region (Vi=VDS<=VGS-VTN). * The total power dissipation is zero just as in region I.          |  | | --- | |  |   Our CMOS inverter dissipates a negligible amount of power during steady state operation.  Power dissipation only occurs during switching and is very low.  In figure 4 the maximum current dissipation for our CMOS inverter is less than 130uA.  Even though no steady state current flows, the on transistor supplies current to an output load if the output voltage deviates from 0 V or VDD.  This makes CMOS technology useable in low power and high-density applications.      https://courseware.ee.calpoly.edu/~dbraun/courses/ee307/F02/02_Shelley/cmos_files/image008.gif  Figure – Drain Current Verses Input Voltage |

**DAILY ASSESSMENT FORMAT**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Date: | 12-06-20 | Name: | Russell D’souza | | Course: | Beginner PHP and MySQL Tutorial | USN: | 4AL15EC023 | | Topic: | EMAIL with PHP,Real life PHP | Semester & Section: | 8th A | | Github Repository: | Russell1005 |  |  |  |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | |  |  |  |  | |  |  |  |  | | **AFTERNOON SESSION DETAILS** | | | | | **Image of session** | | | | | **Report – Report can be typed or hand written for up to two pages.**  1. EMAIL with PHP  PHP mail is the built in PHP function that is used to send emails from PHP scripts.  The mail function accepts the following parameters;   * Email address * Subject * Message * CC or BC email addresses   + It’s a cost effective way of notifying users on important events.   + Let users contact you via email by providing a contact us form on the website that emails the provided content.   + Developers can use it to receive system errors by email   + You can use it to email your newsletter subscribers.   + You can use it to send password reset links to users who forget their passwords   + You can use it to email activation/confirmation links. This is useful when registering users and verifying their email addresses   **Why/When to use the mail PHP**  **Sending mail using PHP**  The PHP mail function has the following basic syntax  <?php  mail($to\_email\_address,$subject,$message,[$headers],[$parameters]);  ?>    HERE,   * “$to\_email\_address” is the email address of the mail recipient * “$subject” is the email subject * “$message” is the message to be sent. * “[$headers]” is optional, it can be used to include information such as CC, BCC   + CC is the acronym for carbon copy. It’s used when you want to send a copy to an interested person i.e. a complaint email sent to a company can also be sent as CC to the complaints board.   + BCC is the acronym for blind carbon copy. It is similar to CC. The email addresses included in the BCC section will not be shown to the other recipients.   **Simple Mail Transmission Protocol (SMTP)**  PHP mailer uses Simple Mail Transmission Protocol (SMTP) to send mail.  On a hosted server, the SMTP settings would have already been set.  The SMTP mail settings can be configured from “php.ini” file in the PHP installation folder.  Configuring SMTP settings on your localhost Assuming you are using xampp on windows, locate the “php.ini” in the directory “C:\xampp\php”.   * Open it using notepad or any text editor. We will use notepad in this example. Click on the edit menu   [How to MAIL using PHP or Phpmailer](https://www.guru99.com/images/2013/04/php_ini.png)   * Click on Find… menu   [How to MAIL using PHP or Phpmailer](https://www.guru99.com/images/2013/04/find.png)   * The find dialog menu will appear   [How to MAIL using PHP or Phpmailer](https://www.guru99.com/images/2013/04/find2.png)   * Click on Find Next button   [How to MAIL using PHP or Phpmailer](https://www.guru99.com/images/2013/04/php_ini2.png)   * Locate the entries   + *[mail function]*   + *; XAMPP:* Don’t remove the semi column if you want to work with an SMTP Server like Mercury   + ; SMTP = localhost   + ; smtp\_port = 25   + Remove the semi colons before SMTP and smtp\_port and set the SMTP to your smtp server and the port to your smtp port. Your settings should look as follows     - SMTP = smtp.example.com     - smtp\_port = 25     - *Note the SMTP settings can be gotten from your web hosting providers.*     - If the server requires authentication, then add the following lines.       * auth\_username = [example\_username@example.com](mailto:example_username@example.com)       * auth\_password = example\_password       * Save the new changes.       * Restart[Apache](https://www.guru99.com/apache.html)server.   **Php Mail Example**  Let’s now look at an example that sends a simple mail.  <?php  $to\_email = 'name @ company . com';  $subject = 'Testing PHP Mail';  $message = 'This mail is sent using the PHP mail function';  $headers = 'From: noreply @ company . com';  mail($to\_email,$subject,$message,$headers);  ?>  **Output:**  <https://www.guru99.com/images/1/phpemail_1.png>  **Sanitizing email user inputs**  The above example uses hard coded values in the source code for the email address and other details for simplicity.  Let’s assume you have to create a contact us form for users fill in the details and then submit.   * Users can accidently or intentional inject code in the headers which can result in sending spam mail * To protect your system from such attacks, you can create a custom function that sanitizes and validates the values before the mail is sent.   Let’s create a custom function that validates and sanitizes the email address using the filter\_var built in function.  Filter\_var function The filter\_var function is used to sanitize and validate the user input data.  It has the following basic syntax.  <?php  filter\_var($field, SANITIZATION TYPE);  ?>    HERE,   * “filter\_var(…)” is the validation and sanitization function * “$field” is the value of the field to be filtered. * “SANITIZATION TYPE” is the type of sanitization to be performed on the field such as;   + **FILTER\_VALIDATE\_EMAIL** – it returns true for valid email addresses and false for invalid email addresses.   + **FILTER\_SANITIZE\_EMAIL**– it removes illegal characters from email addresses. info\@domain.(com) returns [info@domain.com](mailto:info@domain.com).   + **FILTER\_SANITIZE\_URL** – it removes illegal characters from URLs. http://www.example@.comé returns >http://www.example@.com   + **FILTER\_SANITIZE\_STRING**- it removes tags from string values. <b>am bold</b> becomes am bold.   The code below implements uses a custom function to send secure mail.  <?php  function sanitize\_my\_email($field) {  $field = filter\_var($field, FILTER\_SANITIZE\_EMAIL);  if (filter\_var($field, FILTER\_VALIDATE\_EMAIL)) {  return true;  } else {  return false;  }  }  $to\_email = 'name @ company . com';  $subject = 'Testing PHP Mail';  $message = 'This mail is sent using the PHP mail ';  $headers = 'From: noreply @ company. com';  //check if the email address is invalid $secure\_check  $secure\_check = sanitize\_my\_email($to\_email);  if ($secure\_check == false) {  echo "Invalid input";  } else { //send email  mail($to\_email, $subject, $message, $headers);  echo "This email is sent using PHP Mail";  }  ?>  **Output:**  <https://www.guru99.com/images/1/phpemail_2.png>  **Secure Mail**  Emails can be intercepted during transmission by unintended recipients.  This can exposure the contents of the email to unintended recipients.  Secure mail solves this problem by transmitting emails via Hypertext Transfer Protocol Secure (HTTPS).  HTTPS encrypts messages before sending them.  **REAL LIFE PHP**  PHP is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.  What is PHP used for?  . It is typically used on websites to generate web page content dynamically. Use-cases include:   * Websites and web applications (server-side scripting) * Command line scripting * Desktop (GUI) applications   Typically, it is used in the first form to generate web page content dynamically. For example, if you have a blog website, you might write some PHP scripts to retrieve your blog posts from a database and display them. Other uses for PHP scripts include:   * Processing and saving user input from form data * Setting and working with website cookies * Restricting access to certain pages of your website   The largest Social Networking Platform, [Facebook](https://www.facebook.com/) is written using PHP  **How does PHP work?**  All PHP code is executed on a web server only, not on your local computer. For example, if you complete a form on a website and submit it, or click a link to a web page written in PHP, no actual PHP code runs on your computer. Instead, the form data or request for the web page gets sent to a web server to be processed by the PHP scripts. The web server then sends the processed HTML back to you (which is where 'Hypertext Preprocessor' in the name comes from), and your web browser displays the results. For this reason, you cannot see the PHP code of a website, only the resulting HTML that the PHP scripts have produced.  This is illustrated below:  PHP is an interpreted language. This means that when you make changes to your source code you can immediately test these changes, without first needing to compile your source code into binary form. Skipping the compilation step makes the development process much faster.  PHP code is enclosed between the <?php and ?> tags and can then be embedded into HTML.  **PHP Frameworks**  Since writing the whole code for a website is not really practical/feasible for most projects, most developers tend to use frameworks for the web development. The advantage of using a framework is that   * You don't have to reinvent the wheel every time you create a project, a lot of the nuances are already taken care for you * They are usually well-structured so that it helps in the separation of concerns * Most frameworks tend the follow the best practices of the language * A lot of them follow the MVC (Model-View-Controller) pattern so that it separates the presentation layer from logic   Popular frameworks   * [CodeIgniter](https://codeigniter.com/) * [Laravel](https://laravel.com/) * [Symfony](https://symfony.com/) * [Zend](http://www.zend.com/) * [CakePHP](https://cakephp.org/) * [FuelPHP](https://fuelphp.com/) * [Slim](https://www.slimframework.com/) * [Yii 2](https://www.yiiframework.com/)   **Basic Syntax**  PHP scripts can be placed anywhere in a document, and always start with <?php and end with ?>. Also, PHP statements end with a semicolon (;).  Here's a simple script that uses the built-in echo function to output the text "The Best PHP Examples"  <!DOCTYPE html>  <html>  <body>  <h1>Developer News</h1>  <?php echo "The Best PHP Examples"; ?>  </body>  </html>  **The output of that would be:**  Developer News  The Best PHP Examples  **OO Programming in PHP**  Object Oriented is an approach to software development that models application around real world objects such as employees, cars, bank accounts, etc. A class defines the properties and methods of a real world object. An object is an occurrence of a class.  The three basic components of object orientation are;   * Object oriented analysis – functionality of the system * Object oriented designing – architecture of the system * Object oriented programming – implementation of the application   **Object Oriented Programming Principles**  The three major principles of OOP are;   * **Encapsulation** – this is concerned with hiding the implementation details and only exposing the methods. The main purpose of encapsulation is to;   + Reduce software development complexity – by hiding the implementation details and only exposing the operations, using a class becomes easy.   + Protect the internal state of an object – access to the class variables is via methods such as get and set, this makes the class flexible and easy to maintain.   + The internal implementation of the class can be changed without worrying about breaking the code that uses the class. * **Inheritance**– this is concerned with the relationship between classes. The relationship takes the form of a parent and child. The child uses the methods defined in the parent class. The main purpose of inheritance is;   + Re-usability– a number of children, can inherit from the same parent. This is very useful when we have to provide common functionality such as adding, updating and deleting data from the database. * **Polymorphism** – this is concerned with having a single form but many different implementation ways. The main purpose of polymorphism is;   + Simplify maintaining applications and making them more extendable.   **OOPs Concepts in PHP**  PHP is an object oriented scripting language; it supports all of the above principles. The above principles are achieved via;   * **Encapsulation** - via the use of “get” and “set” methods etc. * **Inheritance** - via the use of extends keyword * **Polymorphism** - via the use of implements keyword | | | | |