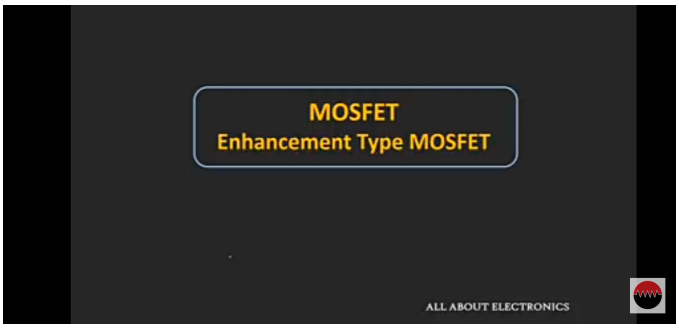
June 9 report

Date:	09/06/2020	Name:	Ankitha c c
Course:	Vlsi	USN:	4al16ec004
Topic:	MOSFET - Enhancement Type MOSFET Explained (Construction, Working and Characteristics Explained)	Semester & Section:	8th & "A" section
Github Repository:	ankitha-c-c		

FORENOON SESSION DETAILS

Image of session



Report – Report can be typed or hand written for up to two pages.

1. MOSFET - Enhancement Type MOSFET Explained (Construction, Working and Characteristics Explained)

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:

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.

Enhancement-type MOSFET or E-MOSFET: The E-MOSFET can be operated only in enhancement mode.

D-MOSFET

Fig.1 shows the constructional detail of n-channel D-MOSFET.

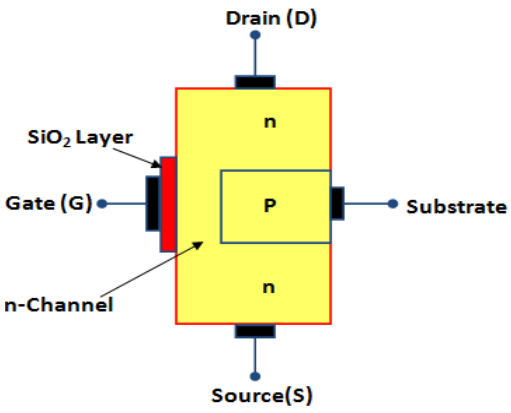


Fig.1 (n-Channel D-MOSFET)

The n-channel D-MOSFET is a piece of n-type material with a p-type region called substrate on the right and an insulated gate on the left as shown in fig.1.

The free electrons flowing from source to drain must pass through the narrow channel between the gate and the p-type region (i.e. substrate).

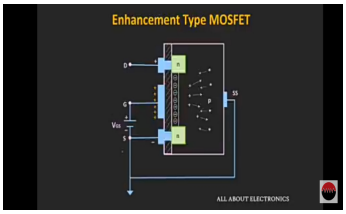
The gate construction of D-MOSFET is explained as below:

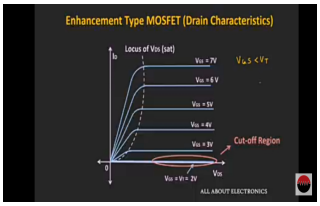
A thin layer of metal oxide, usually silicon dioxide (SiO2) is deposited over a small portion of the channel. A metallic gate is deposited over the oxide layer.

As SiO2 is an insulator, therefore, gate is insulated from the channel.

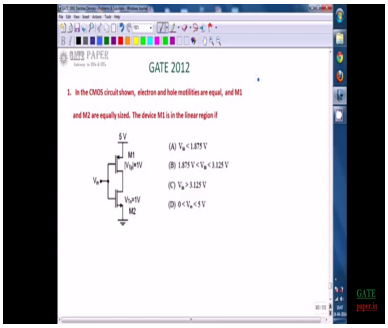
The substrate is connected to the source internally so that a MOSFET has three terminals such as Source (S), Gate (G) and Drain(D).

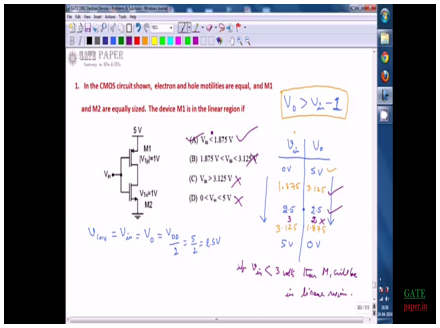
Since the gate is insulated from the channel, we can apply either negative or positive voltage to the gate. Therefore, D-MOSFET can be operated in both depletion-mode and enhancement-mode.



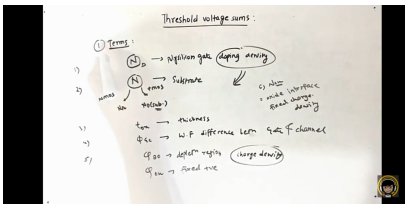


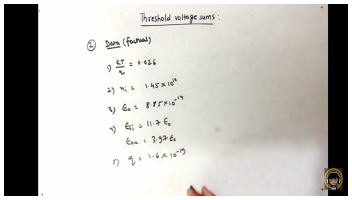
2. GATE 2009 and 20121 ECE operating region and output voltage of CMOS inverter given

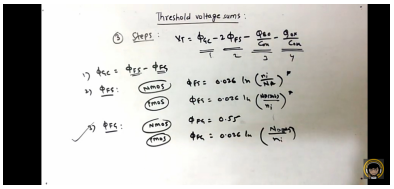


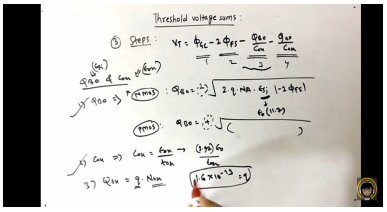


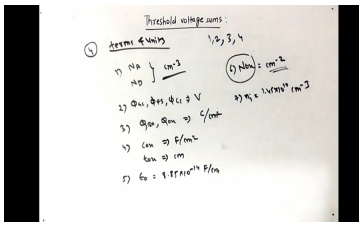
3. MOSFET vth based problems











4. MOSFET problems and solutions

1. In MOSFET devices the N-channel type is better than the P – Channel type in the following respects.

(a) It has better immunity

(b)It is faster

(c) It is TTL compatible

(d)It has better drive capability

Soln. In N – Channel MOSFETs the charge carriers are electrons while in

P – channel MOSFETs holes are the charge carriers.

The mobility of electrons is always greater than the mobility of holes.

i.e. >

Thus, N – Channel MOSFETs are faster

Option (b)

2. In a MOSFET, the polarity of the inversion layer is the same as that of the

(a) Charge on the GATE – EC – electrode

(b)Minority carries in the drain

(c) Majority carriers in the substrate

(d)Majority carries in the source

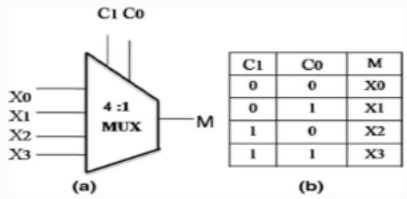
Soln. In a MOSFET the polarity of inversion layer is the same as that of majority carriers in the source. For example, for N – MOSFETs the source is of N – type and inversion layer formed is of electrons.

Option (d)

5. TRICK to implement 4:1 mux using TRANSMISSION GATE & PASS TRANSISTOR LOGIC

It quite often happens, in the design of large-scale digital systems, that a single line is required to carry two or more different digital signals. Of course, only one signal at a time can be placed on the one line. What is required is a device that will allow us to select, at different instants, the signal we wish to place on this common line. Such a circuit is referred to as Multiplexer. The graphical symbol and truth table of 4:1 MUX are shown in Fig. 1a, b, respectively. A multiplexer performs the function of selecting the input on any one of 'n' input lines and feeding this input to one output line.

From: High performance, low power 200 Gb/s 4:1 MUX with TGL in 45 nm technology



C1	C0	M
0	0	X0
0	1	X1
1	0	X2
1	1	X3

4:1 MUX: graphical symbol (a), truth table (b)

figure1

4:1 MUX: graphical symbol (a), truth table (b)

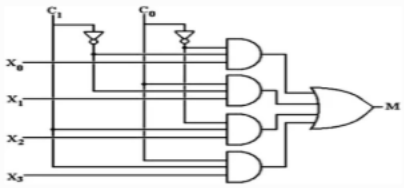
Full size image

Multiplexers are used as one method of reducing the number of integrated circuit packages required by a particular circuit design. This in turn reduces the cost of the system.

Output=X0·C^{----}0·C^{----}1+X1·C^{----}0·C1+X2·C0·C^{----}1+X3·C0·C1

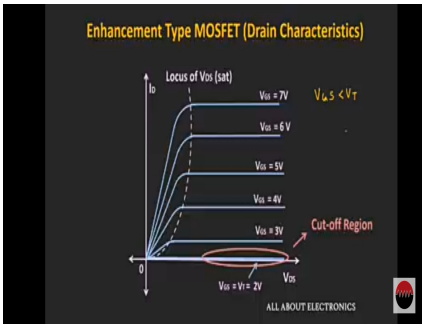
Assume that we have four lines, X0, X1, X2 and X3, which are to be multiplexed on a single line, Output (M). The four input lines are also known as the Data Inputs. Since there are four inputs, we will need two additional inputs to multiplexer, known as the Select Inputs, to select which of the X inputs is to appear at the output, called as select lines C0 and C1. The gate implementation of a 4:1 MUX is shown in Fig. 2. Equation 1 is given for 4:1 MUX.

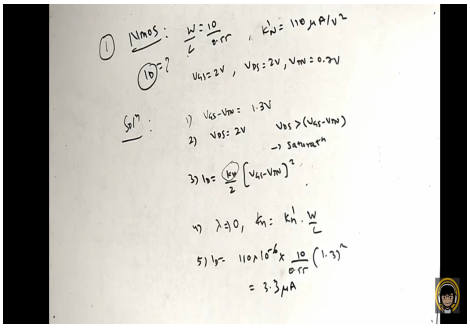
From: High performance, low power 200 Gb/s 4:1 MUX with TGL in 45 nm technology



Gate implementation of a 4:1 Multiplexer

6. MOSFET Drain current - graph , formulae & sums (cutoff,linear& saturation)

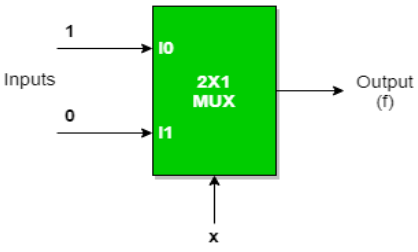




7. Realization of logic function using Multiplexer

a) Implementation of NOT gate using 2 : 1 Mux

NOT Gate :



Truth Table

x	f
0	1
1	0

We can analyze it

$Y = x'.1 + x.0 = x'$

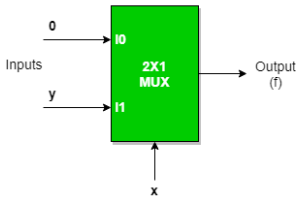
It is NOT Gate using 2:1 MUX.

The implementation of NOT gate is done using "n" selection lines. It cannot be implemented using "n-1"

selection lines. Only NOT gate cannot be implemented using "n-1" selection lines.

b) Implementation of AND gate using 2 : 1 Mux

AND GATE



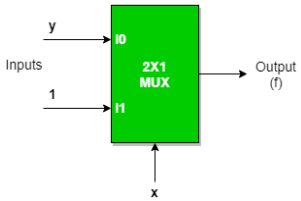
Truth Table

x	y	f	f→y
0	0	0	f = 0
0	1	0	
1	0	0	f = y
1	1	1	

This implementation is done using "n-1" selection lines.

c) Implementation of OR gate using 2 : 1 Mux using "n-1" selection lines.

OR GATE



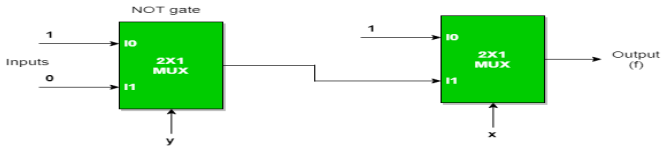
Truth Table

x	y	f	f→y
0	0	0	f = y
0	1	1	
1	0	1	f = 1
1	1	1	

Implementation of NAND, NOR, XOR and XNOR gates requires two 2:1 Mux. First multiplexer will act as NOT gate which will provide complemented input to the second multiplexer.

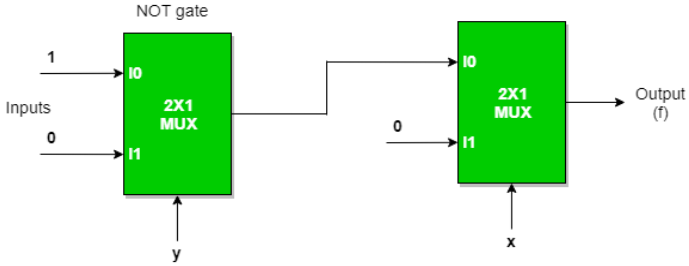
d) Implementation of NAND gate using 2 : 1 Mux

NAND GATE



e) Implementation of NOR gate using 2 : 1 Mux

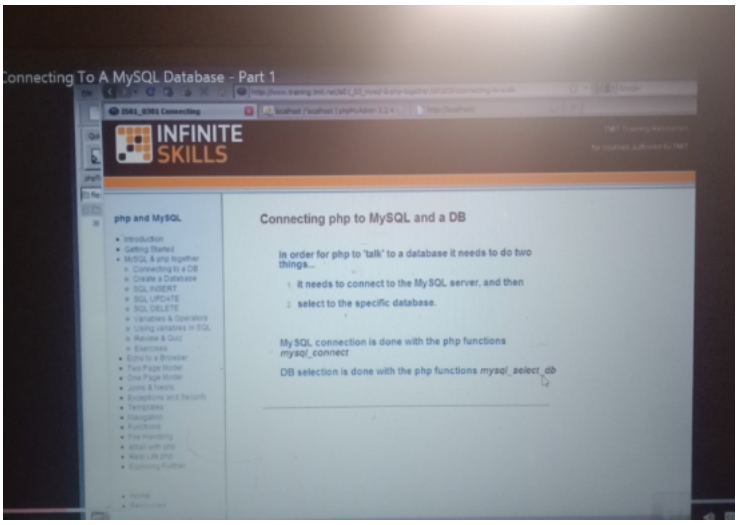
NOR GATE

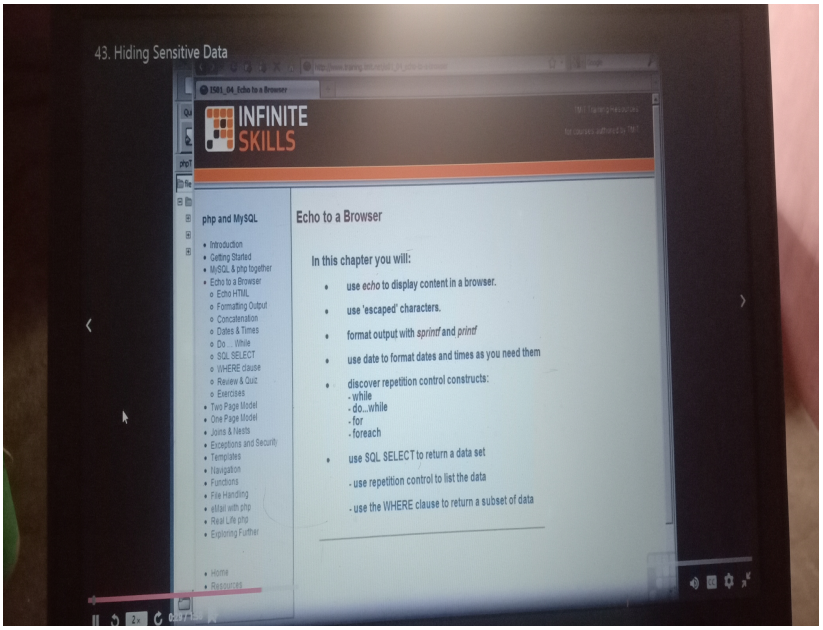


Date:	09/06/2020	Name:	Ankitha c c
Course:	Mysql	USN:	4al16ec004
Topic:		Semester & Section:	8th & a section

AFTERNOON SESSION DETAILS

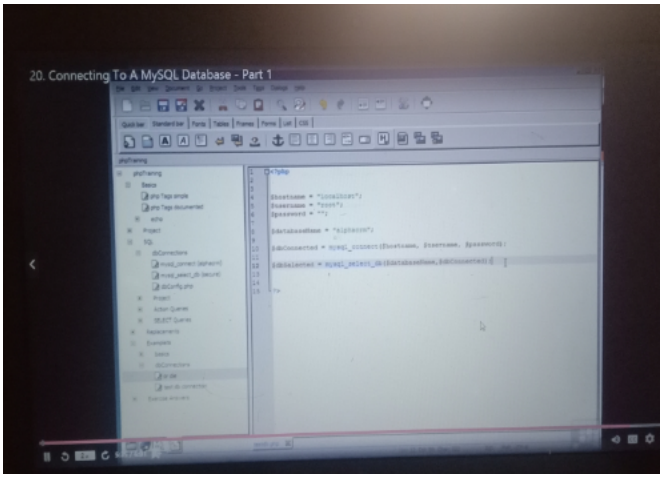
Image of session





Report – Report can be typed or hand written for up to two pages.

1. Out first look at MySQL and PHP



Now that you've seen a bit of the power both of PHP and MySQL, it's time to bring these two juggernauts together. With many programming languages, anytime you want to talk to a database, you have to download and install extra code, or install little plug-in models that give your programs support for talking to that database. PHP isn't like that though; it comes ready to connect to MySQL from the moment you run the php command.

Even though you've only recently begun your journey to PHP mastery, you're ready to use a database from your scripts. You just need to learn a few new commands and how to deal with the problems that can come up when you're working with a database. In fact, you're going to build a simple form that lets you enter SQL and run it against your MySQL database. When you're a PHP programmer you can go beyond the mysql command-line tool.

Then, to put a cherry on top of your towering sundae of PHP and MySQL goodness, you'll write another script. This script will take all the information from the forms you've built, add that information to a database, and then add one more form to let your users search for another user by name. All that in one chapter? Yes indeed.

Writing a Simple PHP Connection Script

No matter how simple or advanced your PHP scripts, if they talk to a database, they'll begin with the same few steps:

Connect to a MySQL installation.

USE the right MySQL database.

Send SQL to the database.

Get the results back.

Do something with the results.

Steps 3, 4, and 5 will change depending on what you're doing. A script that creates tables looks a little different from a script that searches through existing tables.

But those first couple of steps—connecting to MySQL and using the right database—are always the same, no matter how fancy your script. Just think, then: the code you're about to write is the same code that programmers making $150 or

$200 an hour are writing somewhere. (They're just writing that code in much fancier houses with robots serving them iced tea as they lounge by the pool.)

Connecting to a MySQL Database

First, you have to tell your PHP script how to connect to a database. This process is basically telling PHP to do what you did when you started up your MySQL command-line client (MySQL on Mac OS X). When you connected to your web server's database, you probably used a command like the followng:

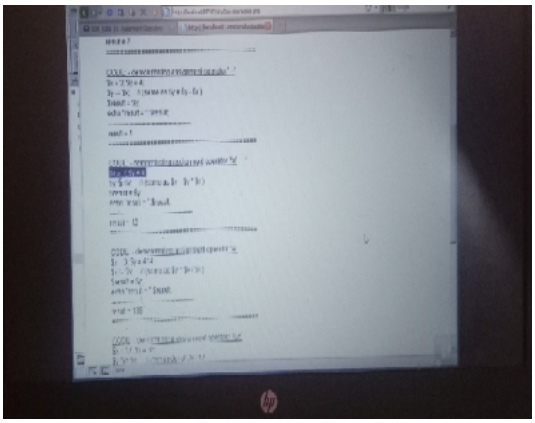
```
bmclaugh@akila:~$ mysql --host=dc2-mysql-02.kattare.com
                    --user=bmclaugh --password
```

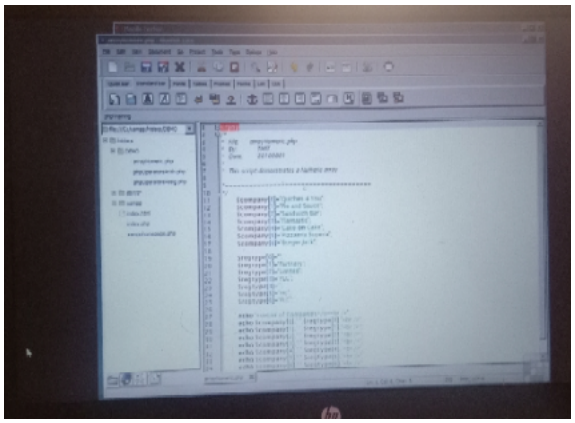
You need those same pieces of information to give PHP so it can connect: your database host, your username, and a password.

Fire up your text editor and create a new script; call it connect.php. This script will be as simple as possible, because all you need it to do is connect to your database, USE the right database, and then run a sample SQL query to make sure things are working correctly.

In your script, type the following lines:

```
<?php
 mysql_connect("your.database.host",
          "your-username", "your-password")
  or die("<p>Error connecting to database: " .
       mysql_error() . "</p>");
 echo "<p>Connected to MySQL!</p>";
```





2. Outputting and processing data

The query and database structure

```
CREATE TABLE `posts` (
  `id` int(11) NOT NULL,
```

```
  `AcceptedAnswerId` int(11) DEFAULT NULL,
  `AnswerCount` int(11) DEFAULT NULL,
  `Body` longtext CHARACTER SET utf8 NOT NULL,
  ...
  `OwnerUserId` int(11) DEFAULT NULL,
  ...
  `Title` varchar(250) CHARACTER SET utf8 DEFAULT NULL,
  `ViewCount` int(11) NOT NULL
  PRIMARY KEY (`Id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1

CREATE TABLE `votes` (
  `Id` int(11) NOT NULL,
  `PostId` int(11) NOT NULL,
  `UserId` int(11) DEFAULT NULL,
  `BountyAmount` int(11) DEFAULT NULL,
  `VoteTypeId` int(11) NOT NULL,
  `CreationDate` datetime NOT NULL,
  PRIMARY KEY (`Id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
SELECT
  v.UserId,
  COUNT(*) AS FavoriteCount
FROM
  Votes v
  JOIN Posts p ON p.id = v.PostId
WHERE
  p.OwnerUserId = 12345678
  AND v.VoteTypeId = 5  -- (Favorites vote)
```

```
GROUP BY
  v.UserId
ORDER BY
  FavoriteCount DESC
LIMIT
  100;
```