**DAILY ASSESSMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| Date: | 09/06/2020 | Name: | GAURAV NR |
| Course: | VLSI | USN: | 4AL15EC025 |
| Topic: | |  | | --- | | 1. MOSFET - Enhancement Type MOSFET Explained (Construction, Working and Characteristics Explained)  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  5. TRICK to implement 4:1 mux using TRANSMISSION GATE & PASS TRANSISTOR LOGIC  6. MOSFET Drain current - graph , formulae & sums (cutoff,linear& saturation)  7. Realization of logic function using Multiplexer | | Semester & Section: | 8TH SEM & A Section |
| Github Repository: | gaurav |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
|  |

|  |
| --- |
| **Report** |
| **Introduction**  **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.  **TYPES**  1. 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.  2. Enhancement-type MOSFET or E-MOSFET: The E-MOSFET can be operated only in enhancement mode.  **GATE 2009 and 20121 ECE operating region and output voltage of CMOS inverter given**      **4-1-multiplexer\_using\_CMOS\_logic | Pass-Transistor-Logic**  4:1 multiplexer using CMOS logic The path selector logic Boolean expression can be given as :  Out = AS + B––S  When the select line signal S is high A is passed to the output and when S is low B is passed to the output. The same logic is used for 4 : 1 MUX in which number of inputs are four (A, B, C, D) and the number of select lines are two (S1, S2). The Boolean expression for 4 : 1 MUX can be given as :  Out = A (S1  S2) + B (S1  ––S2) + C (––S1  S2) + D (––S1  ––S2)  The above Boolean expression can be used to implement 4 : 1 multiplexer or 1 : 4 demultiplexer. The above logic can be generalised as :  2m = n  Where n is the number of inputs in case of MUX (outputs in case of DEMUX) and m is the number of control lines.  4 : 1 MUX using CMOS logic  The implementation of 4 : 1 MUX using CMOS logic is shown in Figure below.  4 : 1 MUX using transmission gates  The implementation of 4 : 1 MUX using transmission gates is shown in  Figure below.    **MOSFET Drain current - graph , formulae & sums (cutoff,linear& saturation)**  In general, any MOSFET is seen to exhibit three operating regions viz.,  1. Cut-Off Region Cut-off region is a region in which the MOSFET will be OFF as there will be no current flow through it. In this region, MOSFET behaves like an open switch and is thus used when they are required to function as electronic switches.  2. Ohmic or Linear Region Ohmic or linear region is a region where in the current IDS increases with an increase in the value of VDS. When MOSFETs are made to operate in this region, they can be used as amplifiers.  3. Saturation Region In saturation region, the MOSFETs have their IDS constant inspite of an increase in VDS and occurs once VDS exceeds the value of pinch-off voltage VP. Under this condition, the device will act like a closed switch through which a saturated value of IDS flows. As a result, this operating region is chosen whenever MOSFETs are required to perform switching operations.    **Realization of logic function using Multiplexer**  It is a combinational circuit which have many data inputs and single output depending on control or select inputs. For N input lines, log n (base2) selection lines, or we can say that for 2n input lines, n selection lines are required. Multiplexers are also known as “Data n selector, parallel to serial convertor, many to one circuit, universal logic circuit”. Multiplexers are mainly used to increase amount of the data that can be sent over the network within certain amount of time and bandwidth    **MOSFET DRAIN GRAPH** |

**DAILY ASSESSMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **09/06/2020** | **Name:** | **GAURAV NR** |
| **Course:** | **Begineer PHP and SQL** | **USN:** | **4AL15EC025** |
| **Topic:** | **Outputting and processing data**  **Dealing with variables**  **Inserting and using database data** | **Semester & Section:** | **8TH SEM & A Section** |
| **Github Repository:** | **gaurav** |  |  |

|  |
| --- |
| **AFTERNOON SESSION DETAILS** |
|  |

|  |
| --- |
| **Report**-  U-SQL does not only support processing unstructured data with the [EXTRACT](https://docs.microsoft.com/en-us/u-sql/statements-and-expressions/extract-expression) expression but also provides the OUTPUT Statement that writes a rowset back into an unstructured file.  The output processing is done in parallel unless otherwise specified. The rowset will be split into parts which then are written into several file parts in parallel which at the end will be stitched together. The degree of parallelism depends on how much data the rowset contains, what the job’s specified degree of parallelism is etc.. Note that the stitching performance depends on the efficiency of the underlying storage system. For example, the stitching of files in the Azure Data Lake is just a meta-data operation and thus very efficient. For more information about the processing model of outputters, please refer to [U-SQL Programmability Guide: User-Defined Outputter](https://docs.microsoft.com/azure/data-lake-analytics/data-lake-analytics-u-sql-programmability-guide#user-defined-outputter).  OUTPUT will provide atomic (all or nothing) semantics. A file is only committed when the script succeeds. If an OUTPUT statement fails for any reason during execution, for example due to a system failure or a user code error in a custom outputter, then the original file (if present) will be preserved and the new file will not be written. Note that this failure will cause the script to fail and any other written file in that script will also be reverted to the before-script status.  OUTPUT will provide snapshot isolation of its writes if the underlying file system provides some form of [multi-version concurrency control](https://en.wikipedia.org/wiki/Multiversion_concurrency_control) (MVCC). This means concurrently executing scripts continue to see the last committed version until the script succeeds. Only jobs started after the script succeeds will see the new file content. syntax Output\_Statement :=  'OUTPUT' [Output\_Rowset](https://docs.microsoft.com/en-us/u-sql/output-statement#out_row)  TO\_Clause  [ [Order\_By\_Opt\_Fetch\_Clause](https://docs.microsoft.com/en-us/u-sql/output-statement#OBOFC) ]  [USING\_Clause](https://docs.microsoft.com/en-us/u-sql/output-statement#us_cla).  TO\_Clause :=  'TO' [Output\_File\_Path](https://docs.microsoft.com/en-us/u-sql/output-statement#out_fp).  Remarks  * Output\_Rowset Specifies the expression that is being written into the target file or files. The supported rowset expressions are any of the following:  Syntax Output\_Rowset :=  Rowset  Rowset\_Expression.  Rowset :=  Rowset\_Variable  |Identifier.  Rowset\_Expression :=  '(' [Query\_Expression](https://docs.microsoft.com/en-us/u-sql/query-statements-and-expressions) ')'  | [Function\_Call](https://docs.microsoft.com/en-us/u-sql/statements-and-expressions/table-valued-function-expression)  | [External\_Rowset\_Expression](https://docs.microsoft.com/en-us/u-sql/statements-and-expressions/select/from/select-selecting-from-an-external-rowset).    The simplest rowset sources are a rowset variable that has been defined in a previous statement of the script or a table that has been created in the account’s catalog. A table can be referenced either with its fully 3-part qualified name, within the current database context with a 2-part name or within the current database and schema context with a single-part name.  Other rowsets that can be output are any query expression inside parenthesis, a table-valued function call or an external rowset expression. Syntax Order\_By\_Opt\_Fetch\_Clause :=  Order\_By\_Clause [ [Offset\_Fetch](https://docs.microsoft.com/en-us/u-sql/statements-and-expressions/order-by-and-offset-fetch-clause" \l "off_F) ].    Because the order by list expressions can only refer to columns of the rowset that needs to be outputted, one can only order on data contained in the result or use an order expression that does not refer to a column at all.  Unlike the [ORDER BY](https://docs.microsoft.com/en-us/u-sql/statements-and-expressions/order-by-and-offset-fetch-clause) clause on the [SELECT](https://docs.microsoft.com/en-us/u-sql/statements-and-expressions/select-expression) expression, the OUTPUT ORDER BY clause does not require the OFFSET/FETCH clause. However, if it is specified it can be used to output only a subset of the data. For further details on the [ORDER BY](https://docs.microsoft.com/en-us/u-sql/statements-and-expressions/order-by-and-offset-fetch-clause) and OFFSET/FETCH syntax and semantics see [ORDER BY and OFFSET/FETCH Clauses (U-SQL)](https://docs.microsoft.com/en-us/u-sql/statements-and-expressions/order-by-and-offset-fetch-clause).   * USING\_Clause The USING clause specifies which outputter should be used to turn the rowset into a file.  Syntax USING\_Clause :=  'USING' udo\_expression.    It takes a C# expression that returns an instance of IOutputter. U-SQL provides a small set of predefined outputters for common text formats and users can write their own by implementing an IOutputter (see [U-SQL Programmability Guide: User-Defined Outputter](https://docs.microsoft.com/azure/data-lake-analytics/data-lake-analytics-u-sql-programmability-guide#user-defined-outputter) for more detail on how to write your own outputter). The built-in outputters are part of the built-in Outputters namespace. |