**Experiment / Assignment / Tutorial No. \_\_\_1\_\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Batch: B2 Roll No.: 1611103 Experiment / assignment / tutorial No.: 1** |

|  |
| --- |
| **Title:** Basic Gates & Universal Gates |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** To study the basic gates: AND, OR, NOT and universal gates: NAND, NOR, XOR, XNOR

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO1:** Recall basic gates and binary, octal & hexadecimal calculations and conversions.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

* R. P. Jain, “Modern Digital Electronics”, Tata McGraw Hill
* http://www.ee.surrey.ac.uk/Projects/Labview/gatesfunc/
* http://www.electronics-tutorials.ws/boolean/bool\_6.html

**Pre Lab/ Prior Concepts:**

Gate is a logic circuit with one or more inputs but only one output. Gates are digital (two state) circuit because the input & output are either low or high. Gates provide high output for certain combinations of input & for other combinations the output is low. Total number of combinations for a gate is 2^n; where n is number of input.

**Classification:** The two types of gate are:

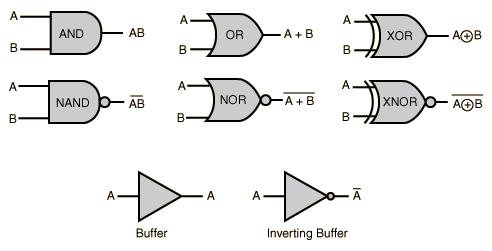
1. **Basic or Fundamental Gates:**

AND, OR & NOT are the basic gates.

1. **Derived Gates:**

NAND, NOR, EX-NOR & EX-OR are the derived gates.

**Symbols of gates**



**Type of IC Symbol**

IC 74266 X-NOR

IC 7486 EX-OR

IC 7402 NOR

IC 7400 NAND

IC 7432 OR

IC 7408 AND

IC 7404 NOT

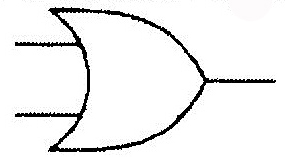
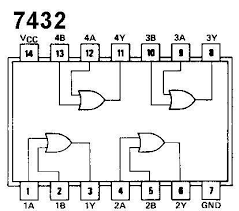
**Implementation Details:**

**Basic Gates**

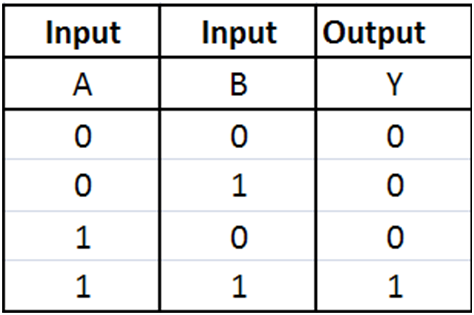
1. **OR gate:** The OR gate has two or more inputs but only 1 output. If any or all the inputs are high, the output is high. If all the inputs are low, the output is low.

Y= A + B

**Symbol for OR gate** **Pin Diagram For IC 7432**

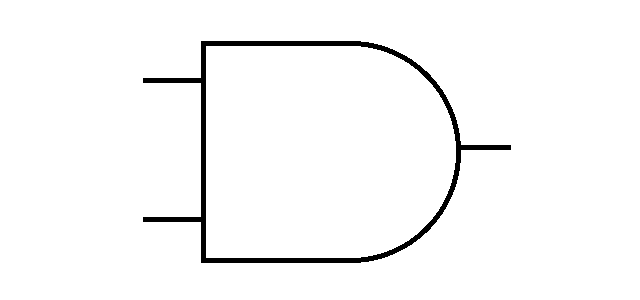
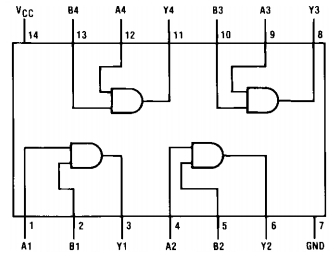
The truth table for OR operations are:



1. **AND gate:** The AND gate has two or more inputs but only one output. If any or all inputs are high then output is also high

Y= A.B

**Symbol for AND gate** **Pin Diagram For IC 7408**

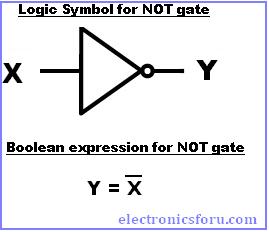
The truth table for AND operations are:

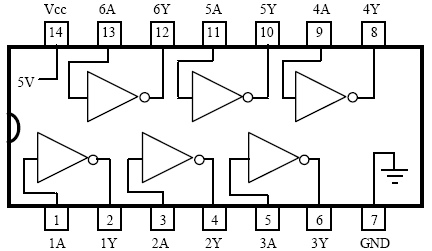


1. **NOT gate:** The Not gate is a gate with only one input and one output. The output is always in opposite state of an input. A NOT gate is also called as Inverter because it performs inversion.

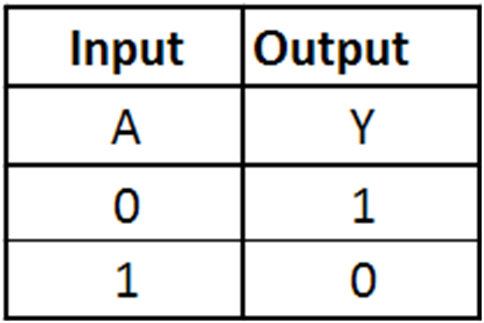
Y=

**Symbol for NOT gate** **Pin Diagram For IC 7404**





The truth table for NOT operations is:



**Derived Gates/Universal Gates**

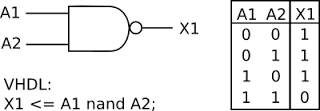
* + NAND gate
  + NOR gate
  + EX-OR gate
  + EX-NOR gate

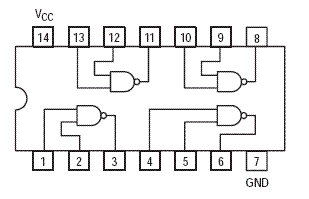
1. **NAND gate:** This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if any of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.

.

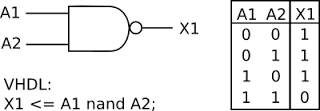
Y= A.B

**Symbol** **Pin Diagram for IC 7400**





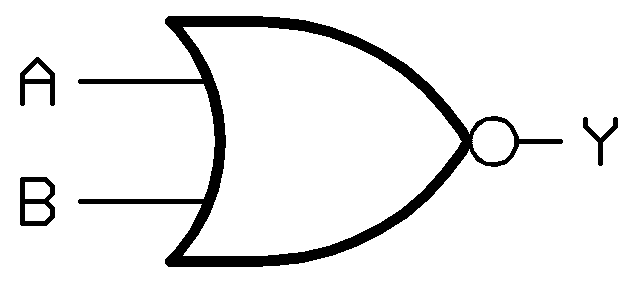
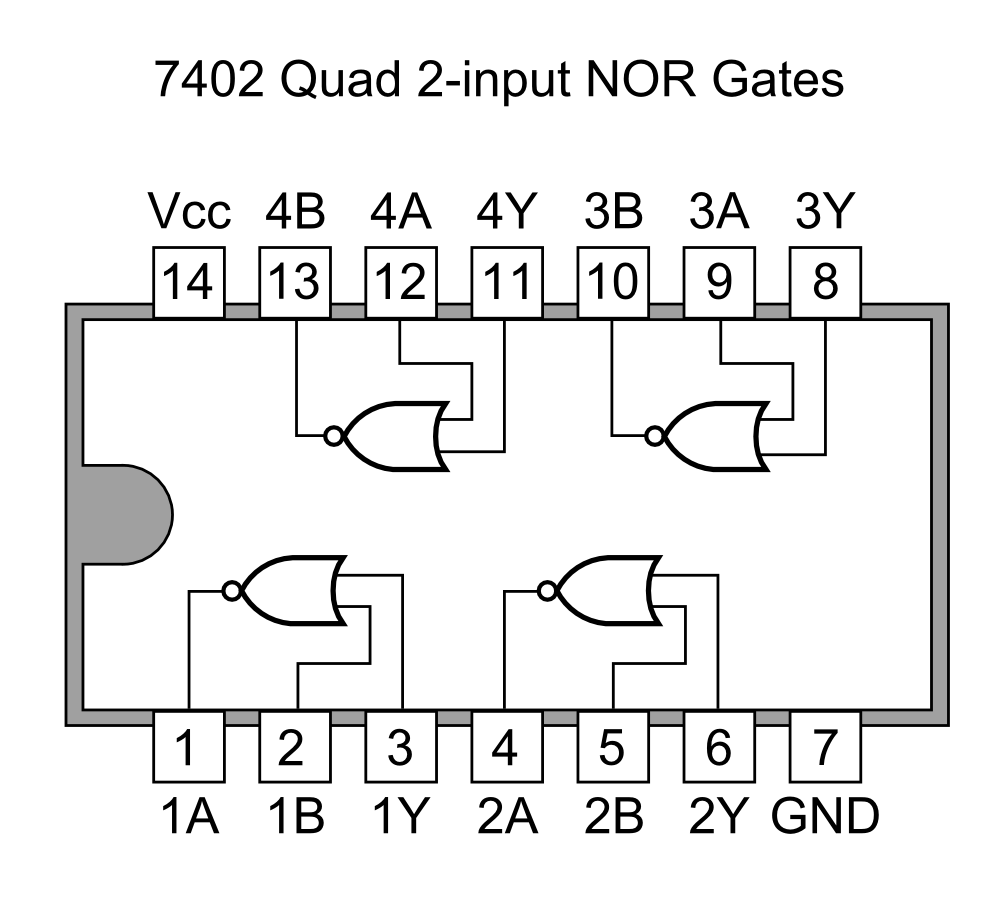
The truth table for NAND operations is:



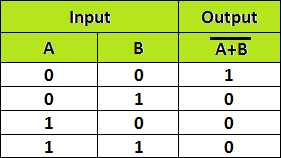
1. **NOR gate:** This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate. The outputs of all NOR gates are low if any of the inputs are high. The symbol is an OR gate with a small circle on the output. The small circle represents inversion.

.

Y= A+B

**Symbol for NOR gate** **Pin Diagram For IC**  

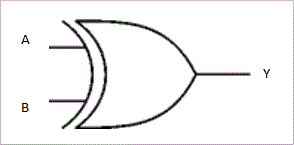
The truth table for NOR operations are:

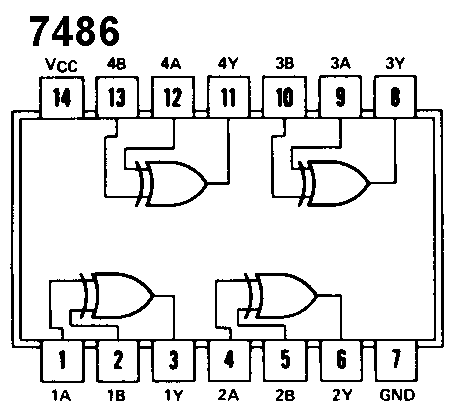


1. **EX-OR gate**: The 'Exclusive-OR' gate is a circuit which will give a high output if either, but not both, of its two inputs are high. An encircled plus sign ( ) is used to show the EX-OR operation

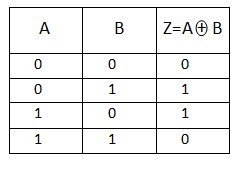
Y= A  B

**Symbol for Ex-OR gate** **Pin Diagram For IC 7486**





The truth table for XOR operations is:

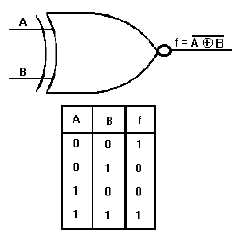


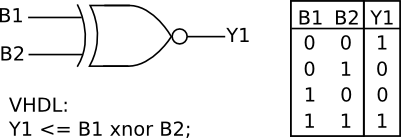
1. **EX-NOR gate**: The 'Exclusive-NOR' gate circuit does the opposite to the EOR gate. It will give a low output if either, but not both, of its two inputs are high. The symbol is an EXOR gate with a small circle on the output. The small circle represents inversion

.

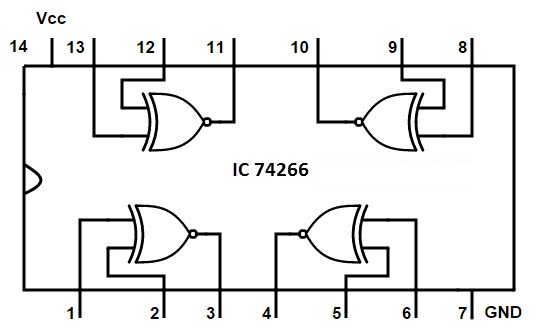
Y= A  B

**Symbol for Ex-NOR gate**

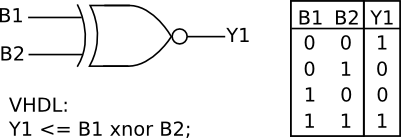




**Pin Diagram for IC 74266**



The truth table for XNOR operations is:

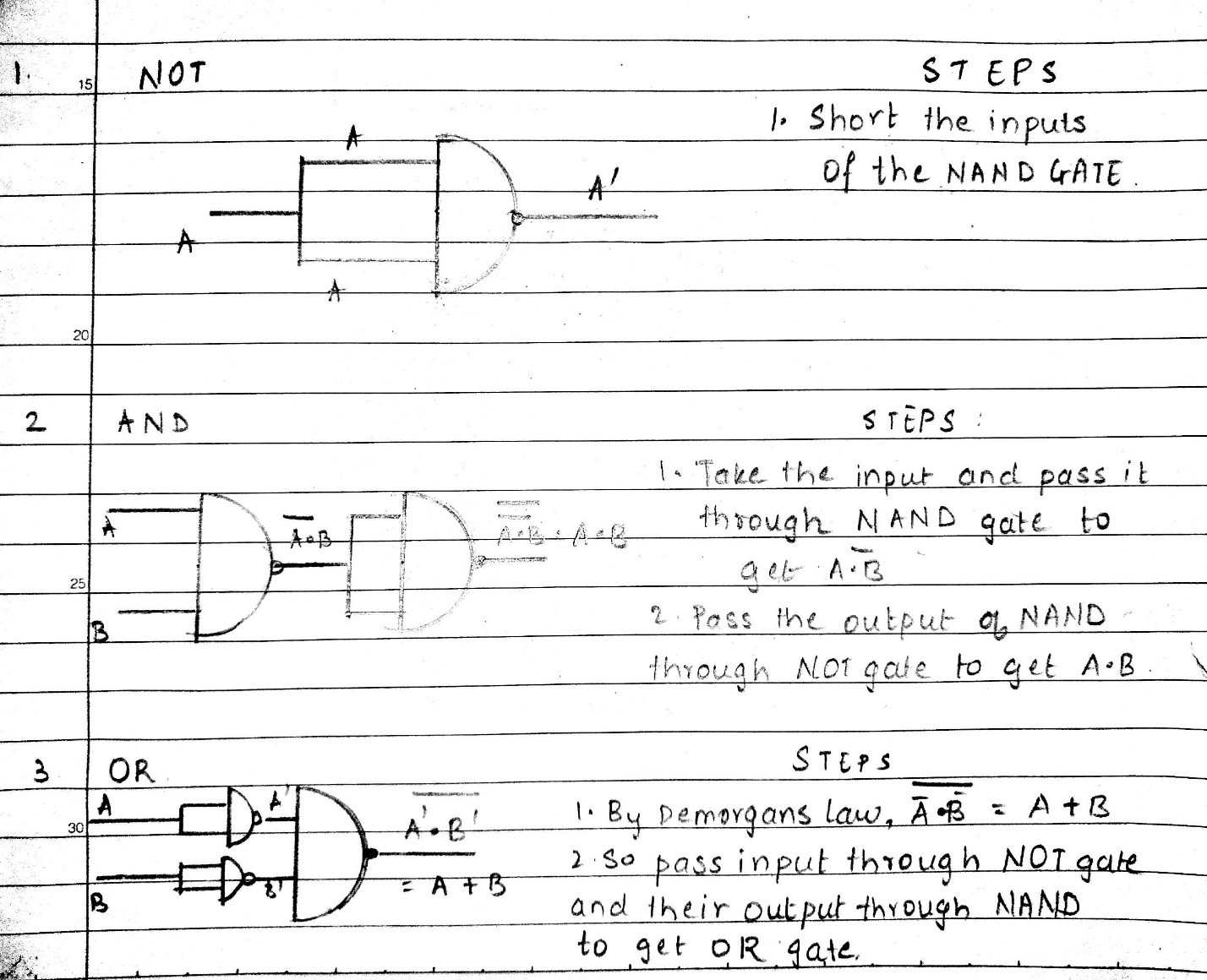


**Implementation Using NAND Gate**

**NOT GATE STEPS**

**AND GATE**

**OR GATE**



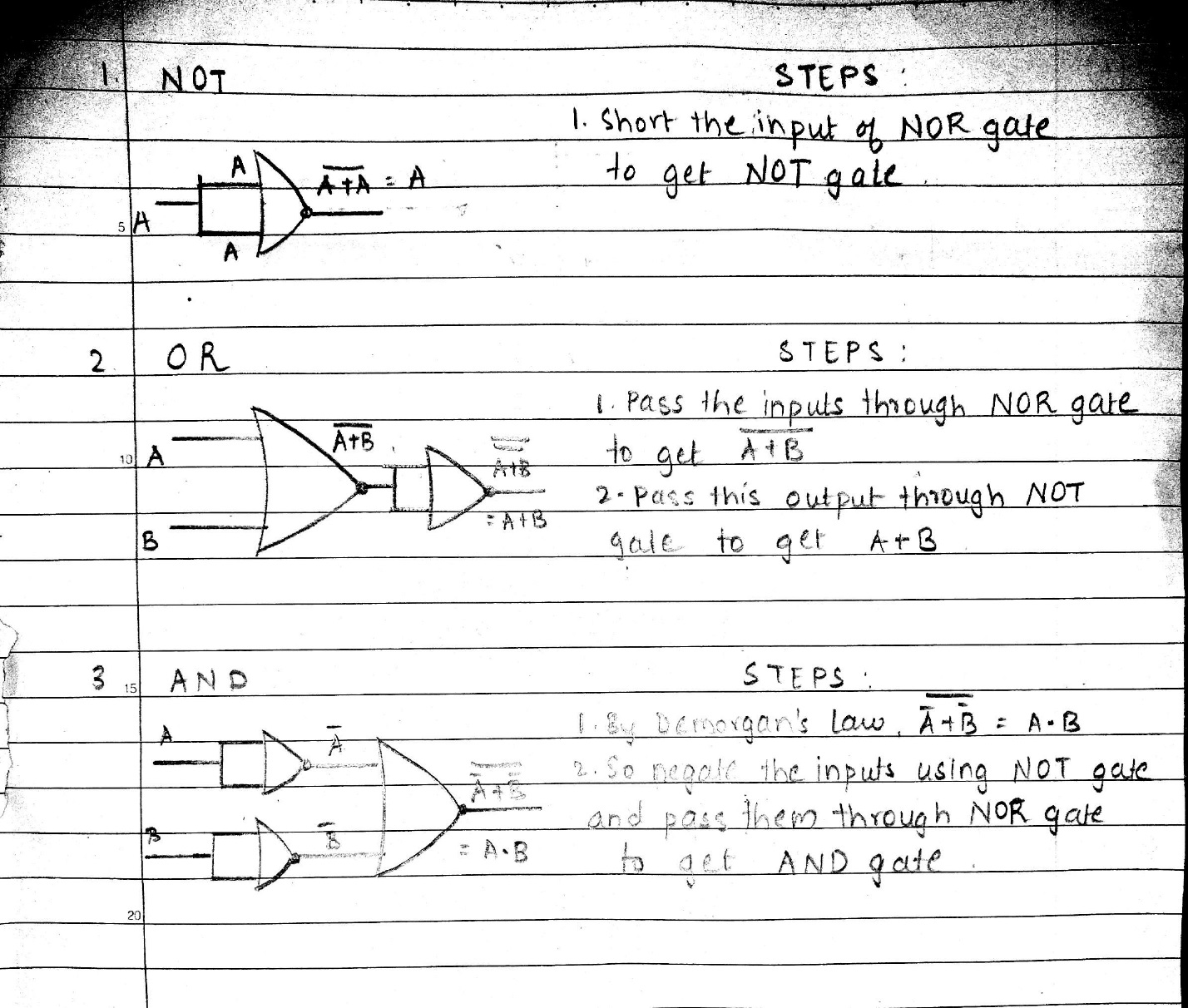
**IMPLEMENTATION USING NOR GATE**

**NOT GATE STEPS**

**OR GATE**

**AND GATE**



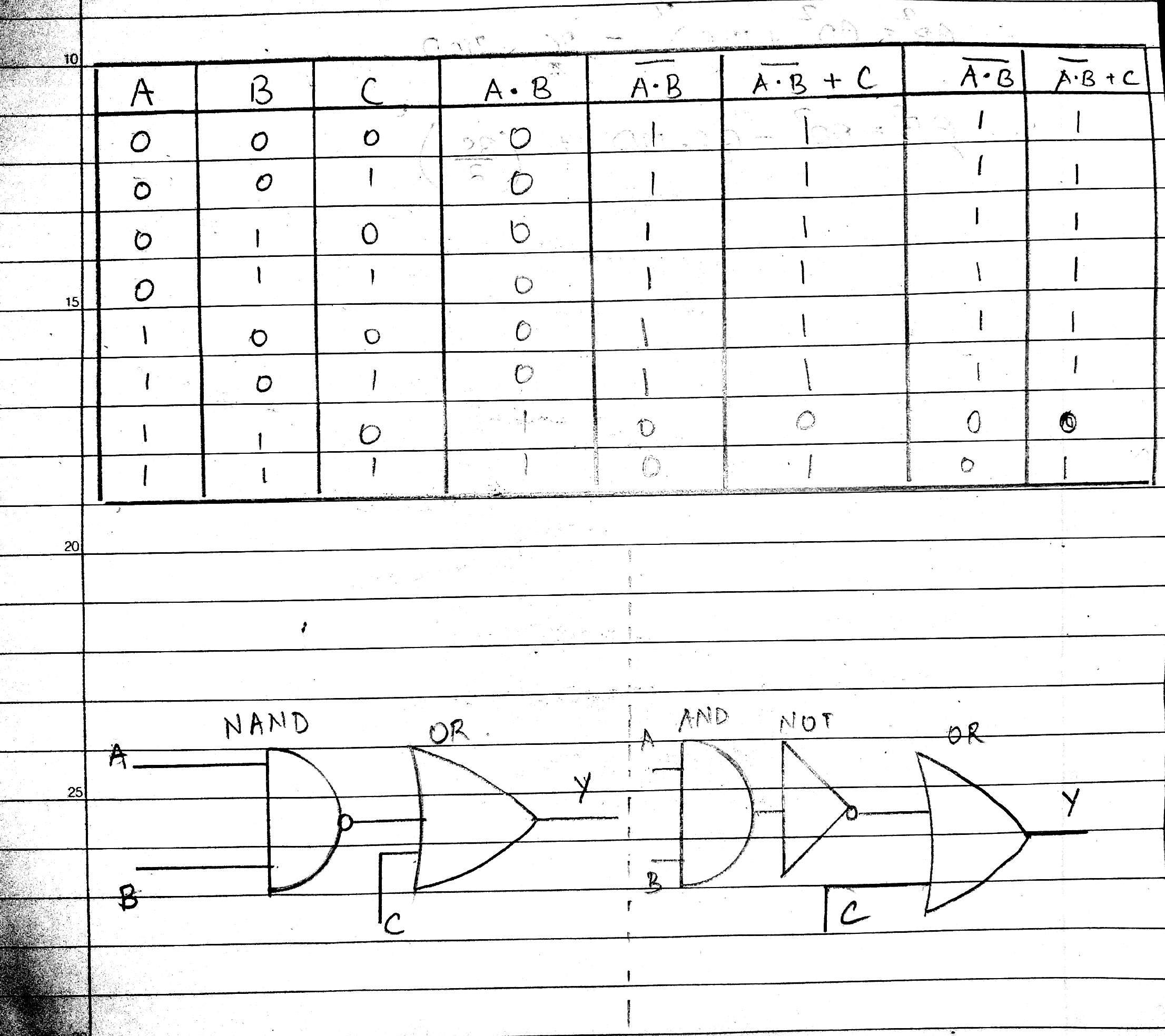


**Conclusion:**

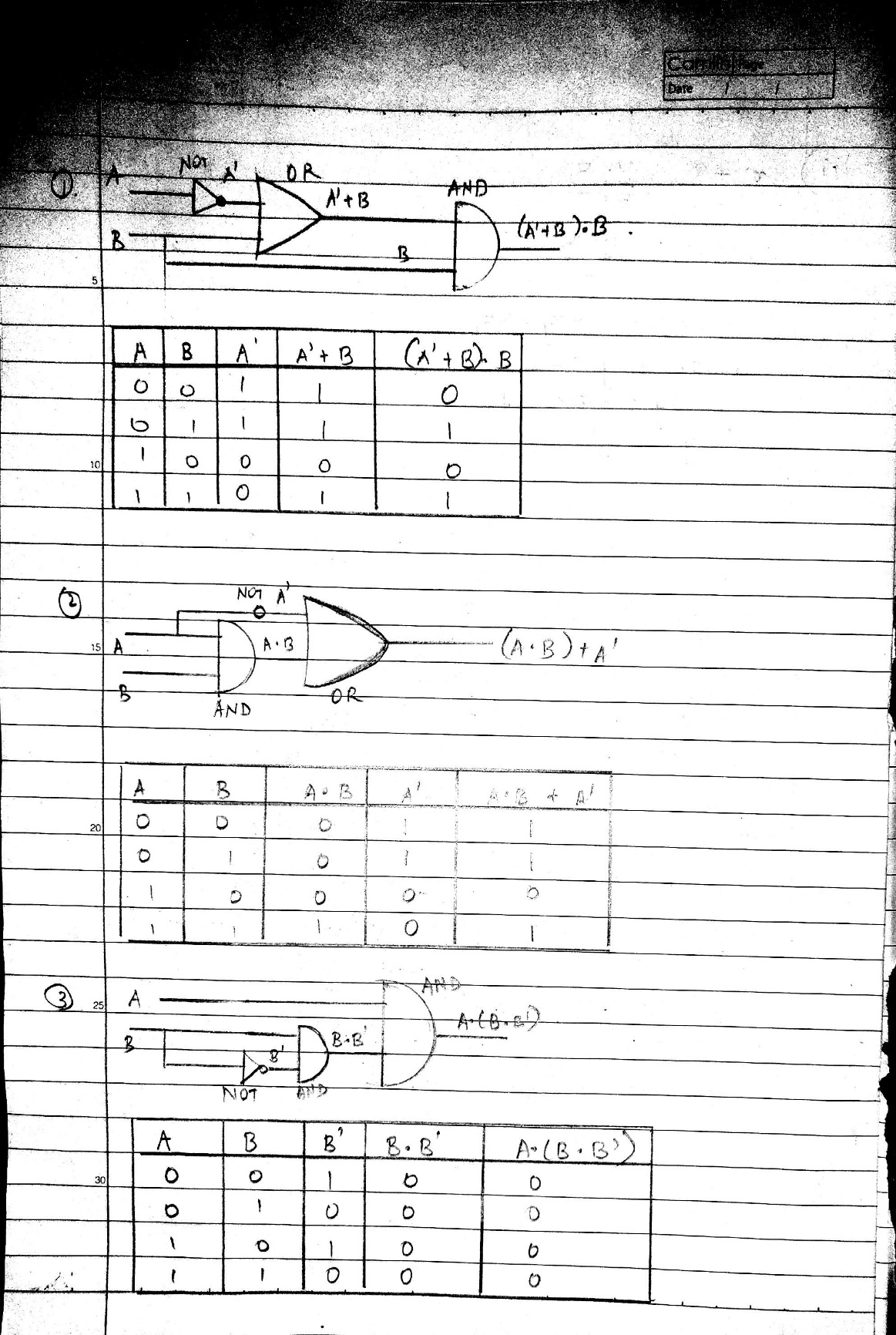
Hence various logic gates and their IC’s were studied and used on electrical PCB and verified.

**Post Lab Descriptive Questions**

1. Verify the expression (A∙B)' + C by:
2. Using NAND Gate directly.
3. Using AND & NOT gate consecutively.



1. Implement the following expressions using combination of gates:
2. (A'+B)∙B
3. (A∙B)+A'
4. A∙ (B∙B')
5. (A'⊕B)∙A



|  |
| --- |
| B**atch: Roll No.: Experiment / assignment / tutorial No.: 2** |

|  |
| --- |
| **Title:** Binary Adders and Subtractors |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** To implement half and full adder–subtractor using gates and IC 7483

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_\_

**Expected Outcome of Experiment:**

**CO2:** Use different minimization technique and solve combinational circuits, synchronous & asynchronous sequential circuits.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

* R. P. Jain, “Modern Digital Electronics”, Tata McGraw Hill
* M .Morris Mano, “Digital Logic & computer Design”, PHI
* http://physics.niser.ac.in/labmanuals/sem5/elect/7\_ADDER%20SUBTRACTO  [R%20CIRCUITS.pd](http://physics.niser.ac.in/labmanuals/sem5/elect/7_ADDER%20SUBTRACTOR%20CIRCUITS.pdf)f

**Pre Lab/ Prior Concepts:**

**Adder:** Addition of two binary digits is most basic operation performed by the digital computer. There are two types of adder:

* Half adder
* Full adder

**Half Adder:** Half adder is combinational logic circuit with two inputs and two outputs. It is the basic building block for addition of two single bit numbers.

**Full adder:** A half adder has a provision not to add a carry coming from the lower order bits when multi bit addition is performed. for this purpose a third input terminal is added and this circuits is to add A,B,C where A and B are the nth order bits of the number A and B respectively and C is the carry generated from the addition of (n-1) order bits. This circuit is referred to as full adder.

**Subtractor:** Subtraction of two binary digits is one of the most basic operations performed by digital computer .there are two types of subtractor:

* Half subtractor
* Full subtractor

**Half subtractor:** Logic circuit for the subtraction of B from A where A,B are 1 bit numbers is referred to as half subtract or .the subtract or process has two input and difference and borrow are the two outputs.

**Full subtractor:** As in the case of the addition using logic gates, a full subtractor is made by combining two half-sub tractors and an additional OR-gate. A full subtractor has the borrow in capability (denoted as BORIN) and so allows cascading which results in the possibility of multi-bit subtraction.

**IC 7483**

For subtraction of one binary number from another, we do so by adding 2’s complement of the former to the latter number using a full adder circuit.

IC 7483 is a 16 pin, 4-bit full adder. This IC has a provision to add the carry output to transfer and end around carry output using Co and C4 respectively.

**2’s complement:** 2’s complement of any binary no. can be obtained by adding 1 in 1’scomplement of that no.

e.g. 2’s complement of +(10)10 =1010is

|  |  |  |  |
| --- | --- | --- | --- |
| 1C of 1010 |  | | 0101 |
|  |  | + | 1 |
| -(10)10 |  | | 0110 |

In 2’s complement subtraction using IC 7483, we are representing negative number in 2’s complement form and then adding it with 1st number.

**Implementation Details:**

**Half Adder Block Diagram**

**Half Adder Circuit**

**Truth Table for Half Adder**

|  |  |  |  |
| --- | --- | --- | --- |
| **Inputs** | | **Outputs** | |
| **A** | **B** | **A** | **B** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**From the truth table (with steps):**

**Full Adder Block Diagram**

**Full Adder Circuit**

**Truth Table for Full Adder**

**From the truth table (with steps):**



**Half Subtractor Block Diagram**

**Half Subtractor Circuit**

**Truth Table for Half Subtractor**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| **A** | **B** | **DIFFERENCE(D)** | **BORROW(Bo)** |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**From the truth table (with steps) :**

**Full Subtractor Block Diagram**

**Full Subtractor Circuit**

**Truth Table for Full subtractor**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **BIN** | **D** | **BOROUT** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**From the truth table (with steps):**

**IC 7483**

**Procedure:**

1. Locate the IC 7483 and 4-not gates block on trainer kit.
2. Connect 1st input no. to A4-A1 input slot and 2nd (negative) no. to B4-B1 through 4-not gates (1C of 2nd no.)
3. Connect high input to Co so that it will get added with 1C of 2nd no. to get 2C.
4. Connect 4-bit output to the output indicators.
5. Switch ON the power supply and monitor the output for various input combinations.

**Example:**

|  |  |  |
| --- | --- | --- |
| 1) 710 -210 = 510 | |  |
| 7 |  | 0111 |
| 2 |  | 0010 |
| 1’C of 2 | | 1101 |
|  |  | + 1 |
| 2’C of 2 | | 1110 |

0111 + 1110 1 0101

**Pin Diagram IC7483**

**Adder**

**Subtractor**

**Conclusion:**

**Post Lab Descriptive Questions**

1. What is difference between half and full adder, half and full subtractor?
2. Perform the following Binary subtraction with the help of appropriate ICs:
3. 7-5
4. 5-7
5. 9-4

**Experiment / Assignment / Tutorial No. \_\_\_\_\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

**Experiment / Assignment / Tutorial No. \_\_3\_\_\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Batch: B2 Roll No.: 1611103 Experiment / assignment / tutorial No.: 3** |

|  |
| --- |
| **Title:** Design 4:1 Multiplexer and 1: 4 De-multiplexer |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** To design and implement a 4:1 multiplexer and 1:4 de-multiplexer using logic gates and MUX IC

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO2:** Use different minimization technique and solve combinational circuits, synchronous & asynchronous sequential circuits.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

* R. P. Jain, “Modern Digital Electronics”, Tata McGraw Hill
* M .Morris Mano, “Digital Logic & computer Design”, PHI
* [https://wiki.engr.illinois.edu/download/attachments/84770821/08](https://wiki.engr.illinois.edu/download/attachments/84770821/08-Multiplexers.pdf?version=2&modificationDate=1285128827000)- [Multiplexers.pdf?version=2&modificationDate=128512882700](https://wiki.engr.illinois.edu/download/attachments/84770821/08-Multiplexers.pdf?version=2&modificationDate=1285128827000)0

**Pre Lab/ Prior Concepts:**

**Multiplexer:** Multiplexer is a special type of combinational circuit. It is a digital circuitwhich selects one of the n data inputs and routes it to the output. The selection of one of the n inputs is done by the select lines. To select n inputs we require m select lines, such that 2m=n. Depending on the digital code applied at the select inputs, one out of the n data sources is selected and transmitted to a single output . E is called as the strobe or enable input which is useful for cascading. It is generally on active low terminal that means it will perform the required operation when it is low. The multiplexer act like a digitally controlled single pole, multiple way switches. The output gets connected to only one input at a time. In most of the electronic system the digital data is available on more than one line. It is necessary to route the data over a single line, under such circumstances input at a time

**Types of Multiplexer:**

1. 2:1 Multiplexer
2. 4:1 Multiplexer
3. 8:1 Multiplexer
4. 16:1 Multiplexer
5. 32:1 Multiplexer

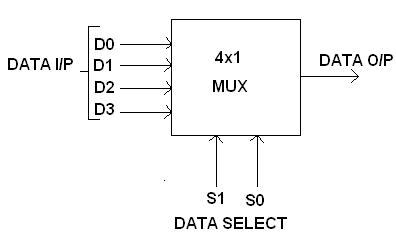
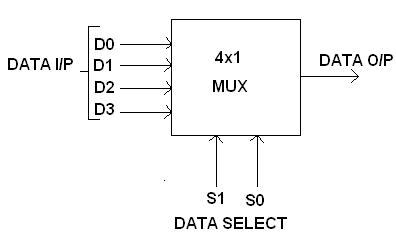
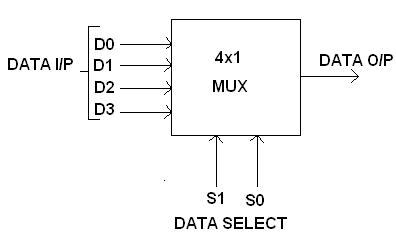
**De-multiplexer:** It has only one input, n output and m select lines. A demultiplexerperforms the reverse operation of a multiplexer i.e. it receives one input and distributes it over several outputs. The demultiplexer converts a serial data signal at the input to a parallel data at its output lines. The relation between the output lines and select lines is as follows: N=2m

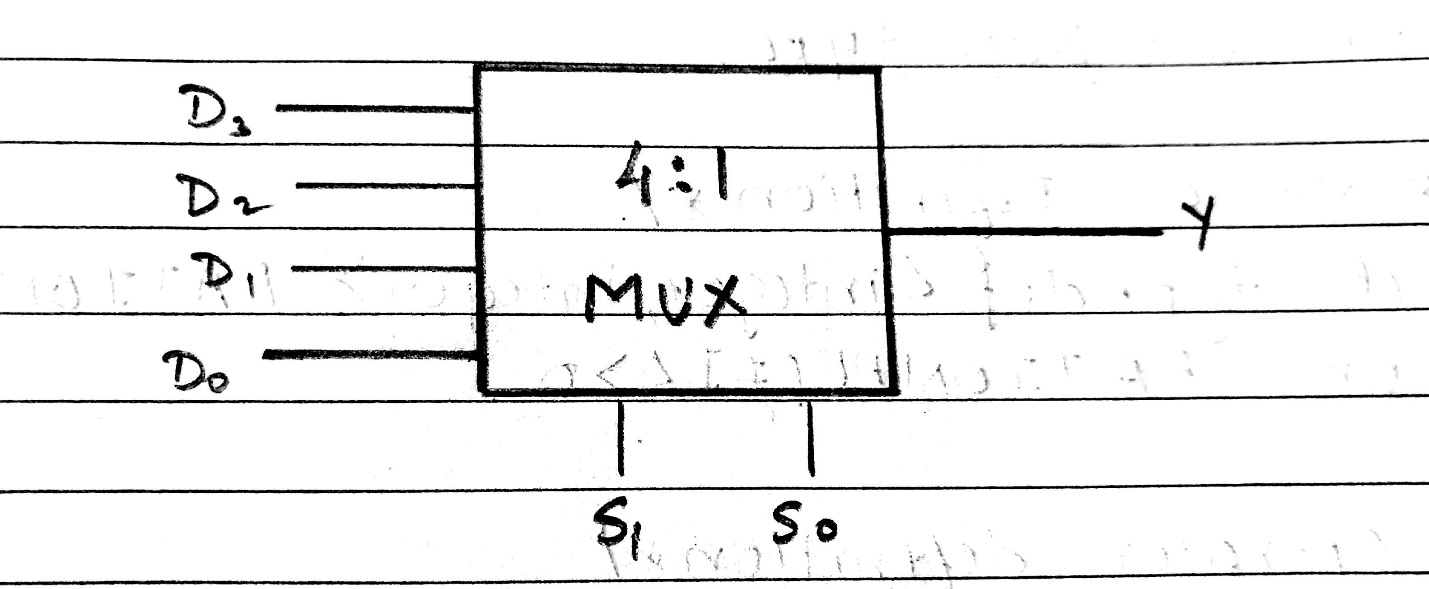
**Types of Demultiplexers:**

1. 1:2 DEMUX
2. 1:4 DEMUX
3. 1:8 DEMUX
4. 1:16 DEMUX

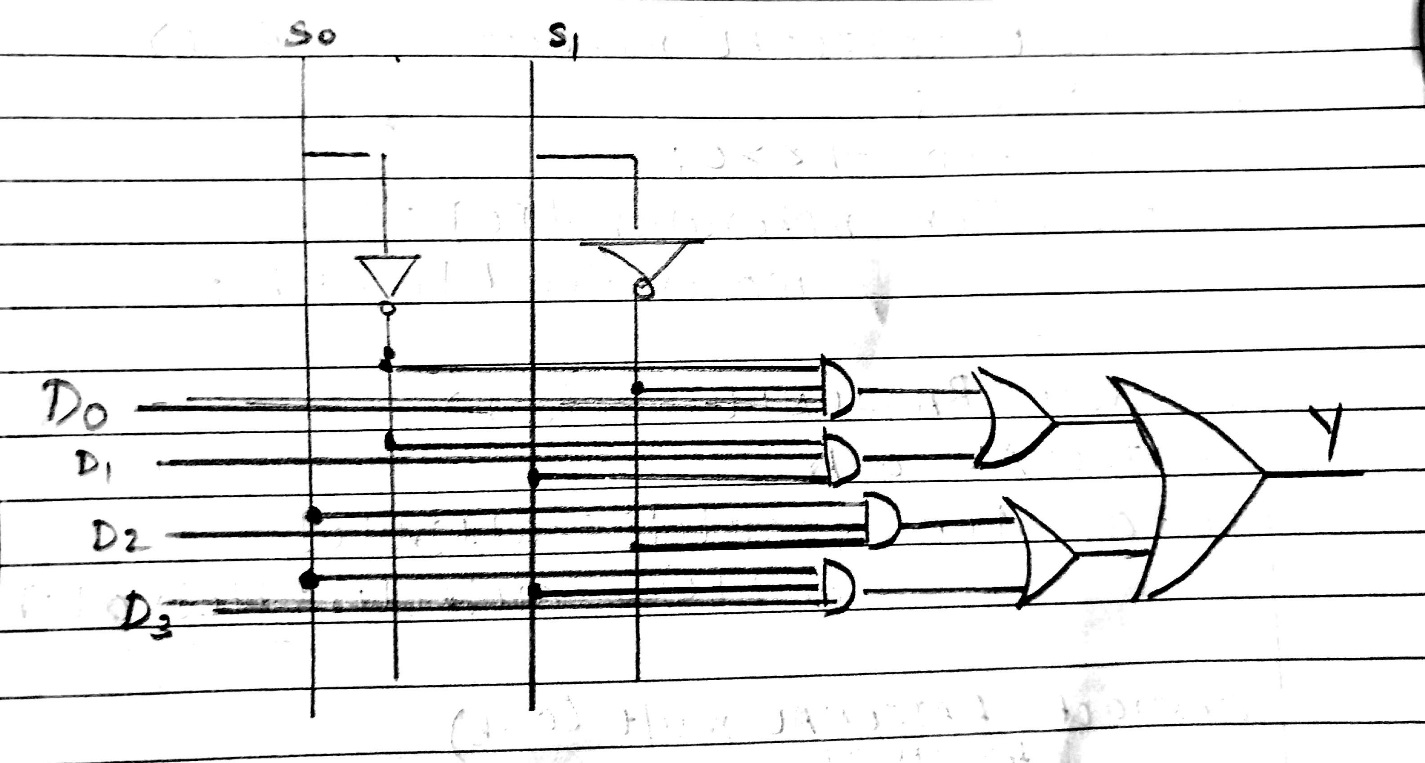
**Implementation Details of 4:1 MUX**

**Block Diagram of 4:1 MUX**





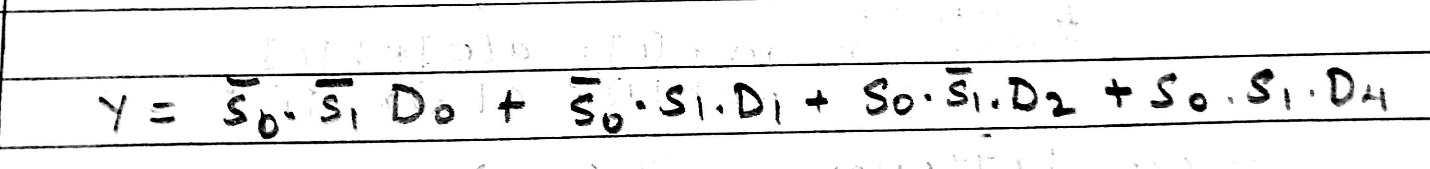
**Circuit Diagram of 4:1 MUX**

****

**Truth table**

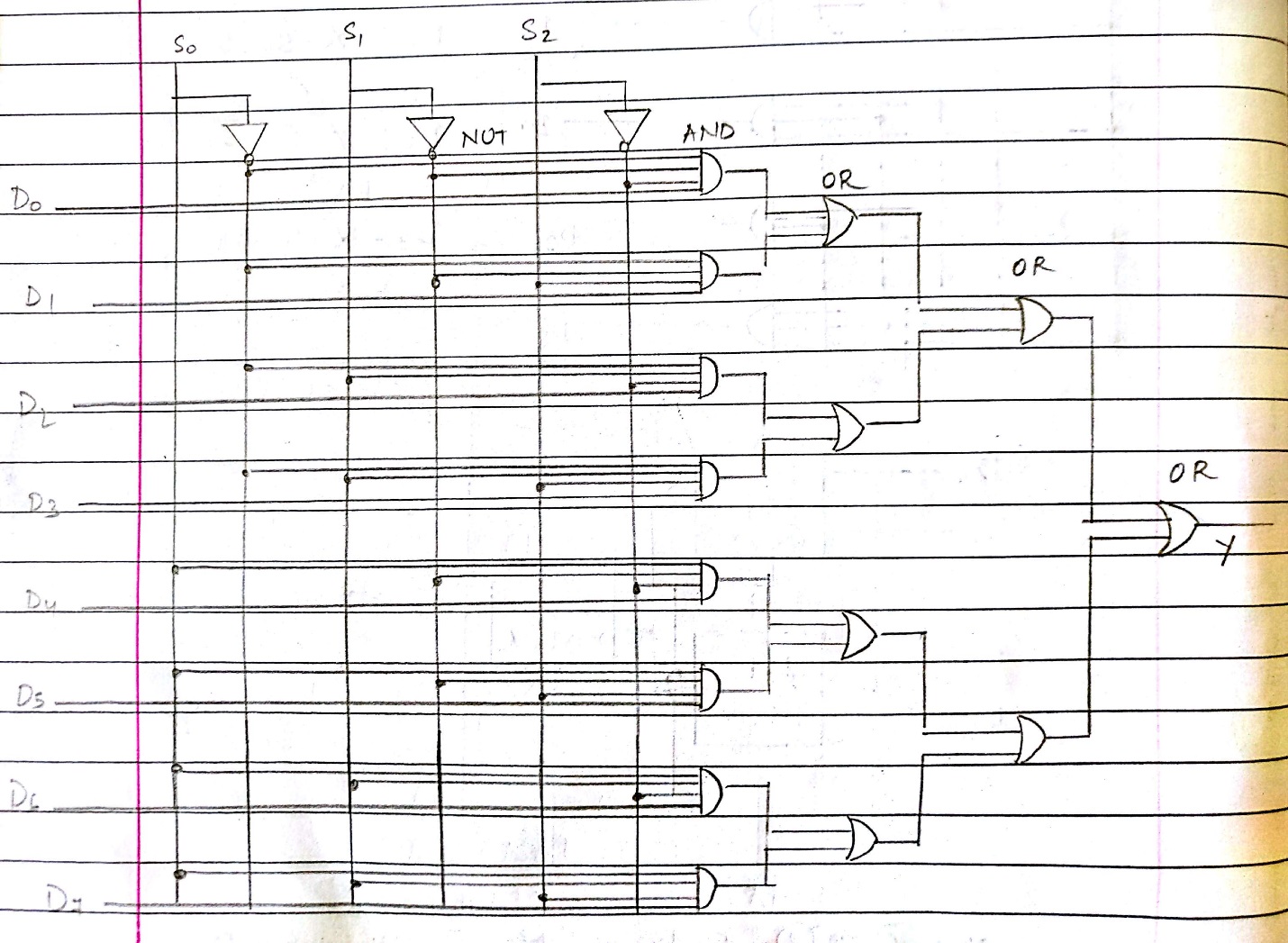
|  |  |  |
| --- | --- | --- |
| **S0** | **S1** | **Y** |
|  |  |  |
|  |  |  |
| 0 | 0 | D0 |
| 0 | 1 | D1 |
|  |  |  |
| 1 | 0 | D2 |
|  |  |  |
|  |  |  |
| 1 | 1 | D3 |
|  |  |  |

**From Truth Table:**

****

**Implementation Details of 8:1 MUX**

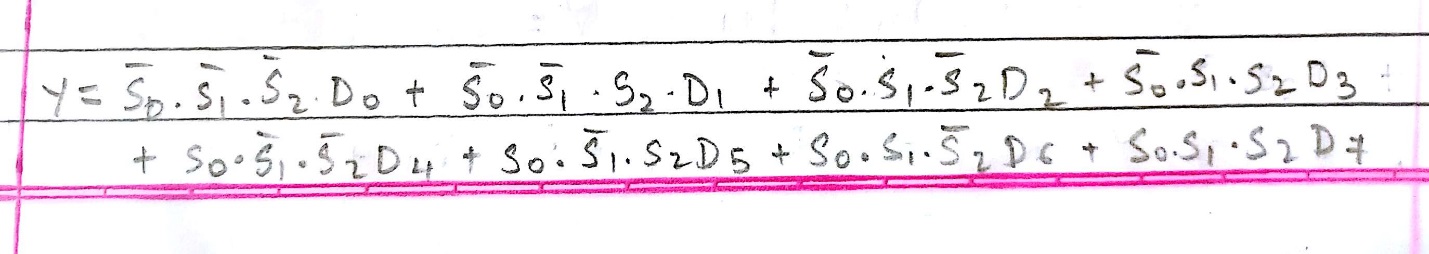
**Circuit Diagram of 8:1 MUX**

****

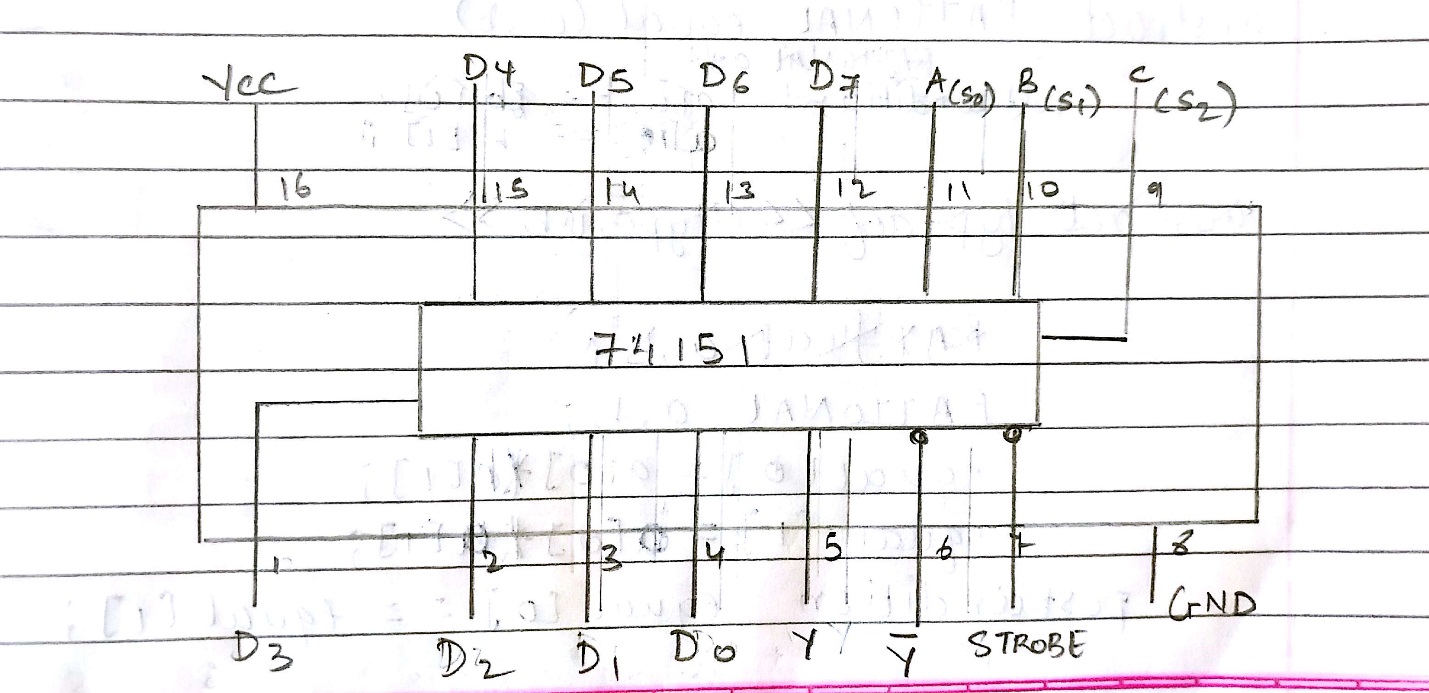
**Truth Table for 8:1 Multiplexer**

|  |  |  |  |
| --- | --- | --- | --- |
| **S0** | **S1** | **S2** | **Y** |
| 0 | 0 | 0 | D0 |
| 0 | 0 | 1 | D1 |
| 0 | 1 | 0 | D2 |
| 0 | 1 | 1 | D3 |
| 1 | 0 | 0 | D4 |
| 1 | 0 | 1 | D5 |
| 1 | 1 | 0 | D6 |
| 1 | 1 | 1 | D7 |

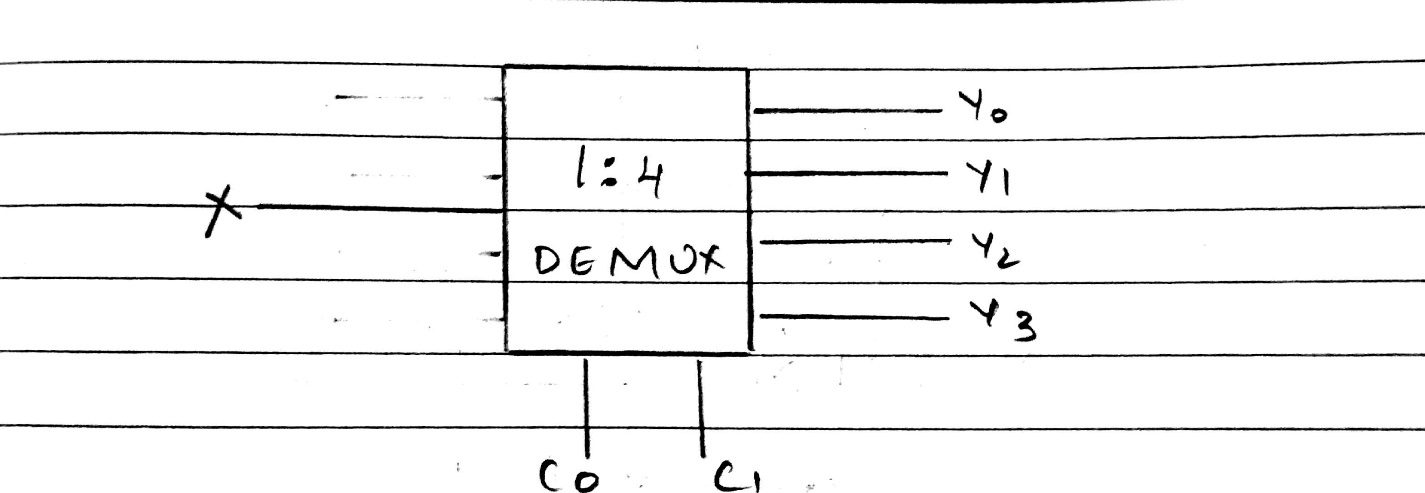
**From Truth Table:**



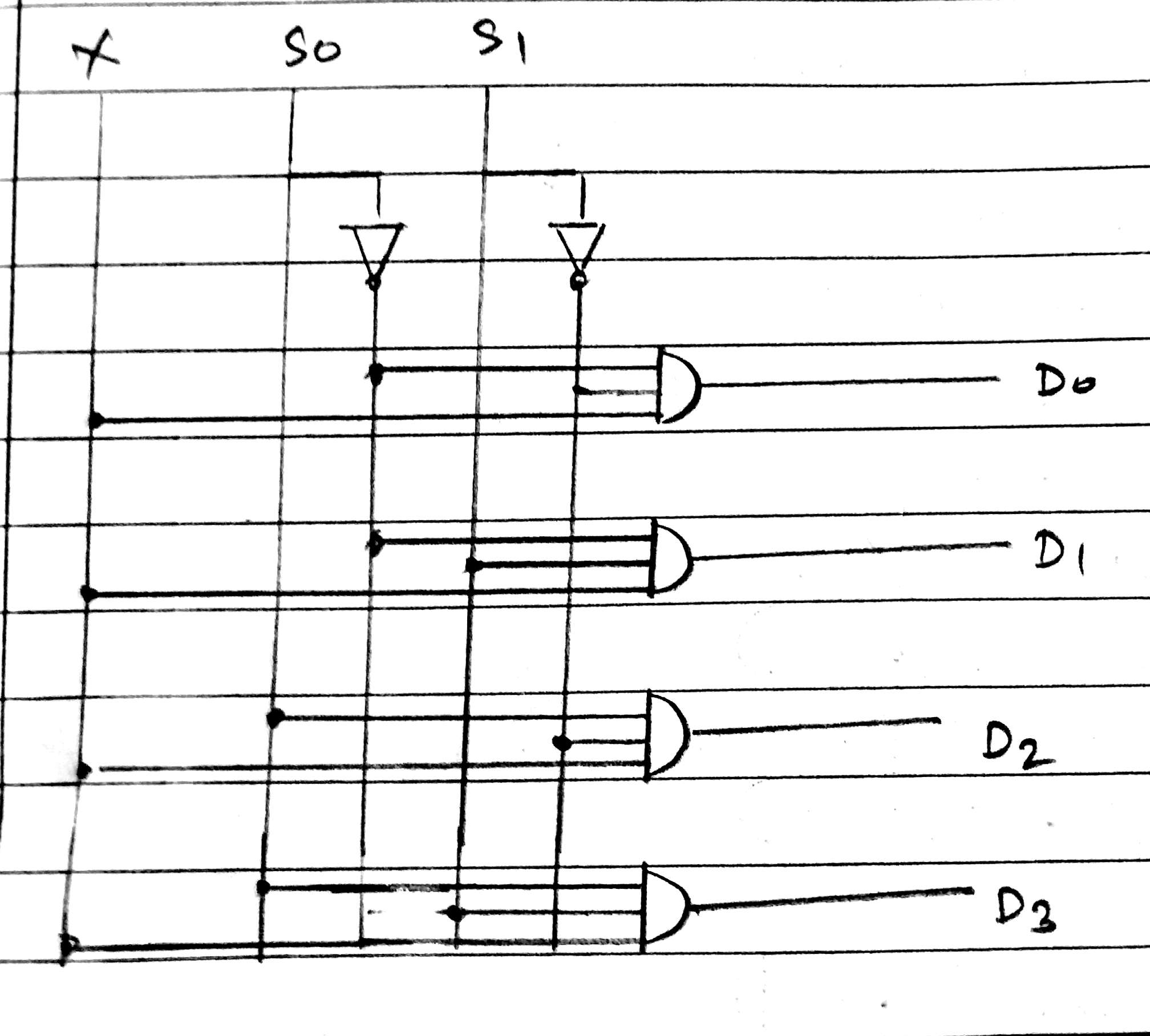
**Pin diagram: IC 74151**



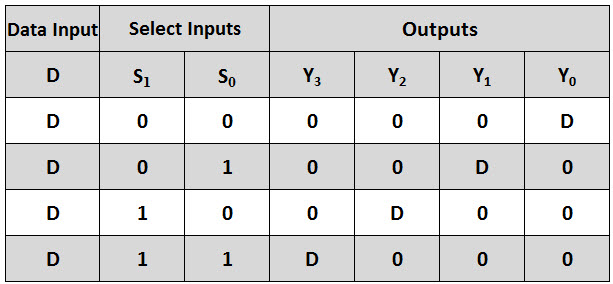
**Block Diagram of 1:4 DE MUX**

****

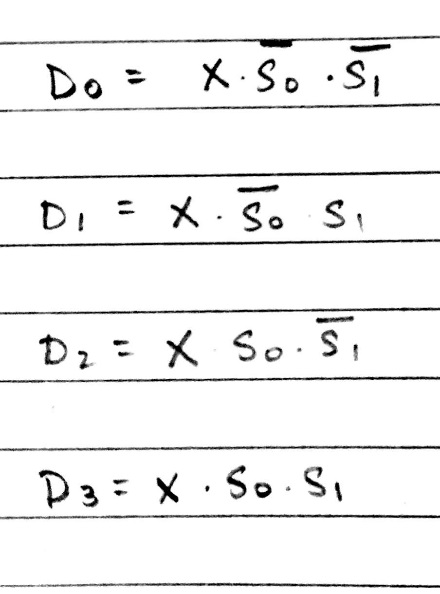
**Circuit Diagram of 1:4 DE MUX**



**Truth Table for 1:4 Demultiplexers**



**From Truth Table:**

****

**Conclusion:**

**Hence the multiplexers and DE multiplexers were studied and their truth tables were verified.**

**Post Lab Descriptive Questions**

1. How many select lines are required for 64:1 MUX?

Input lines = 64.

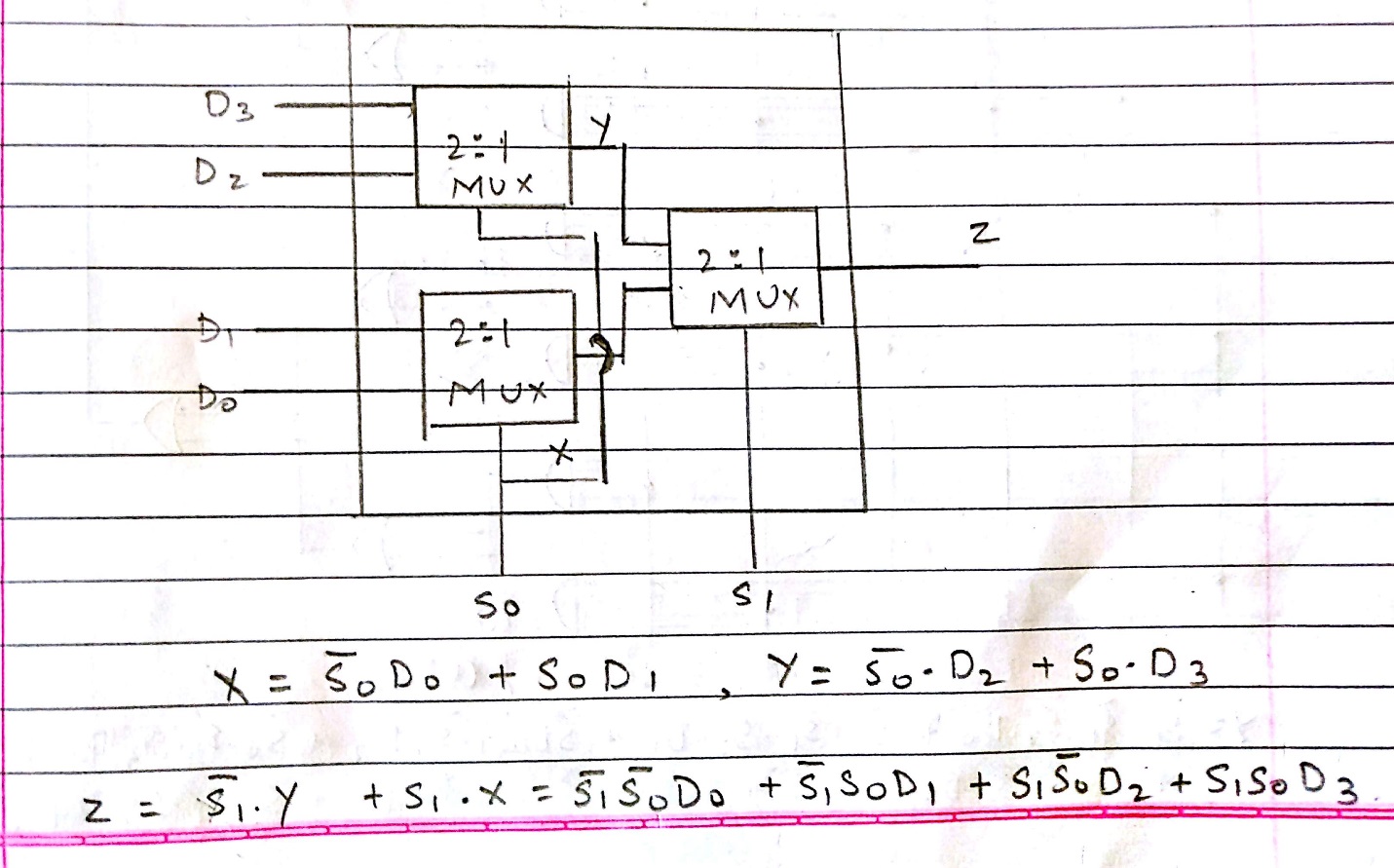
Select line = log2(input lines). (Since input lines = 2(selsect lines)

Therefore Select lines = 6.

1. State some applications of MUX and DEMUX.

Multiplexer are used in various fields where multiple data need to be transmitted using a single line. Following are some of the applications of multiplexers –

1. **Communication system** – Communication system is a set of system that enable communication like transmission system, relay and tributary station, and communication network. The efficiency of communication system can be increased considerably using multiplexer. Multiplexer allow the process of transmitting different type of data such as audio, video at the same time using a single transmission line.
2. **Telephone network –**In telephone network, multiple audio signals are integrated on a single line for transmission with the help of multiplexers. In this way, multiple audio signals can be isolated and eventually, the desire audio signals reach the intended recipients.
3. **Computer** **memory**– Multiplexers are used to implement huge amount of memory into the computer, at the same time reduces the number of copper lines required to connect the memory to other parts of the computer circuit.
4. **Transmission from the computer system of a satellite** – Multiplexer can be used for the transmission of data signals from the computer system of a satellite or spacecraft to the ground system using the GPS (Global Positioning System) satellites.
5. Demultiplexer  is used to connect a single source to multiple destinations. The main application area of demultiplexer is communication system where multiplexer are used. Most of the communication system are bidirectional  i.e. they function in both ways (transmitting and receiving signals). Hence, for most of the applications, the multiplexer and demultiplexer work in sync. Demultiplexer are also used for reconstruction  of parallel data and ALU circuits.
6. **Communication System**– Communication system use multiplexer to carry multiple data like audio, video and other form of data using a single line for transmission. This process make the transmission easier.  The demultiplexer receive the output signals of the multiplexer and converts them back to the original form of the data at the receiving end. The multiplexer and demultiplexer work together to carry out the process of transmission and reception of data in communication system.
7. **ALU (Arithmetic Logic Unit)** – In an ALU circuit, the output of ALU can be stored in multiple registers or storage units with the help of demultiplexer. The output of ALU is fed as the data input to the demultiplexer. Each output of demultiplexer is connected to multiple register which can be stored in the registers.
8. **Serial to parallel** **converter**– A serial to parallel converter is used for reconstructing parallel data from incoming serial data stream.  In this technique, serial data from the incoming serial data stream is given as data input to the demultiplexer at the regular intervals. A counter is attach to the control input of the demultiplexer. This counter directs the data signal to the output of the demultiplexer where these data signals are stored. When all data signals have been stored, the output of the demultiplexer can be retrieved and read out in parallel.
9. Build a 4:1 MUX using only 2:1 MUX.



**Experiment / Assignment / Tutorial No. \_\_\_\_\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Batch: Roll No.: Experiment / assignment / tutorial No.: 4** |

|  |
| --- |
| **Title:** 4 bit Magnitude Comparator |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** Design a 2-bit comparator using logic gates and verify 4-bit magnitudecomparator using IC 7485

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO2:** Use different minimization technique and solve combinational circuits, synchronous & asynchronous sequential circuits.

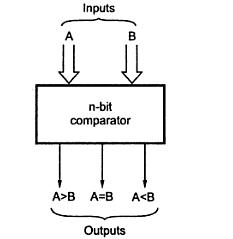
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

* R. P. Jain, “Modern Digital Electronics”, Tata McGraw Hill
* M .Morris Mano, “Digital Logic & computer Design”, PHI
* http://elnsite.teilam.gr/ebooks/digital\_design/lab/dataSheets\_page/7485.pdf

**Pre Lab/ Prior Concepts:**

The comparison of two numbers is an operator that determines one number is greater than, less than (or) equal to the other number. A magnitude comparator is a combinational circuit that compares two numbers A and B and determines their relative magnitude. The outcome of the comparator is specified by three binary variables that indicate whether A>B, A=B (or) A<B.



**Two Bit Magnitude Comparator Implementation Details:**

**Truth Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A1** | **A0** | **B1** | **B0** | **A > B** | **A = B** | **A < B** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**From the Truth Table:**

**(A<B)=**

**(A=B)=**

**(A>B)=**

**Logic Diagram of 2 bit Comparator**

**Four Bit Magnitude Comparator Implementation Details**

**Pin Diagram of IC 7485**



**Logic Diagram of IC 7485**

**Comparing Table**

**Conclusion:**

**Post Lab Descriptive Questions**

1. Design a 1- bit magnitude comparator using logic gates.

**Experiment / Assignment / Tutorial No. \_\_\_\_\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Batch: B2 Roll No.: 1611103 Experiment / assignment / tutorial No.: 5** |

|  |
| --- |
| **Title:** Flip Flops |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:**Design of JK Flip flop, D flip flop, T flip flop using NAND Gates & verification of the same flip flop using IC7476

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO2:** Use different minimization technique and solve combinational circuits, synchronous & asynchronous sequential circuits.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

* R. P. Jain, “Modern Digital Electronics”, Tata McGraw Hill
* M .Morris Mano, “Digital Logic & computer Design”, PHI
* A.P.Godse, D.A.Godse, “Digital Logic Design”

**Pre Lab/ Prior Concepts:**

Flip-flop is the common name given to two-state devices which offer basic memory for sequential logic operations. Flip-flops are heavily used for digital data storage and transfer and are commonly used in banks called "registers" for the storage of binary numerical data.

**JK-flip flop:** has two inputs, traditionally labeled J and K. IC 7476 is a dual JK master slave flip flop with preset and clear inputs. If J and K are different then the output Q takes the value of J at the next clock edge. If J and K are both low then no change occurs. If J and K are both high at the clock edge, then the output will toggle from one state to the other. It can perform the functions of the set/reset flip-flop and has the advantage that there are no ambiguous states.

**D Flip Flop:** tracks the input, making transitions with match those of the input D. The D stands for "data"; this flip-flop stores the value that is on the data line. It can be thought of as a basic memory cell. D flip-flop can be made from J-K flip-flop by connecting both inputs through a not gate.

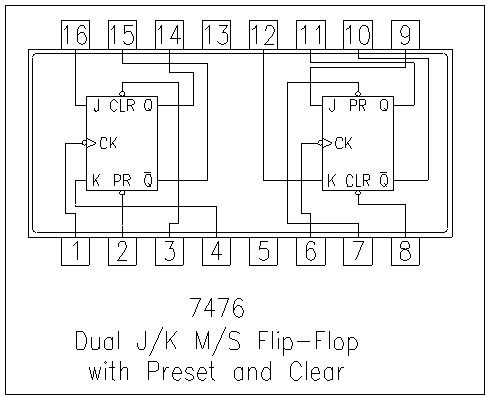
**T Flip Flop:** T or "toggle" flip-flop changes its output on each clock edge, giving an output which is half the frequency of the signal to the T input. It is useful for constructing binary counters, frequency dividers, and general binary addition devices. It can be made from a J-K flip-flop by tying both of its inputs high.

**Implementation Details:**

**Procedure**

1. Locate IC 7476 on Digital trainer kit
2. Apply various inputs to J & K pins by means of the output on logic output indicator.
3. Connect a pulsar switch to the clock input.
4. Connect the J&K as D and T flip flop as shown in diagrams and verify the respective truth tables.

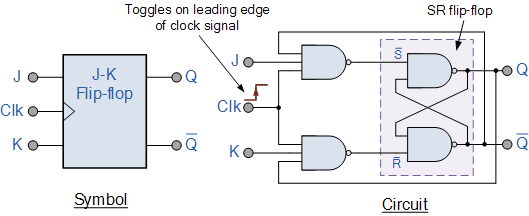
**Pin Diagram of IC 7476 JK Master- Slave FF**



**Logic Symbol Truth Table**

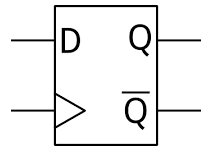
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **J** | **K** | **Q** | **Q’** | **Qn+1** | **Qn+1’**  **n+1** |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | **1** |

**JKFF**

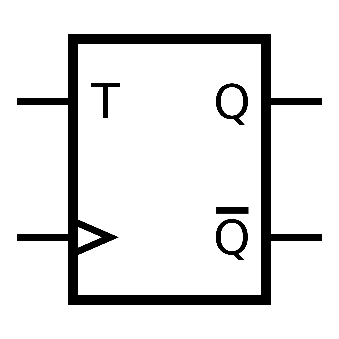


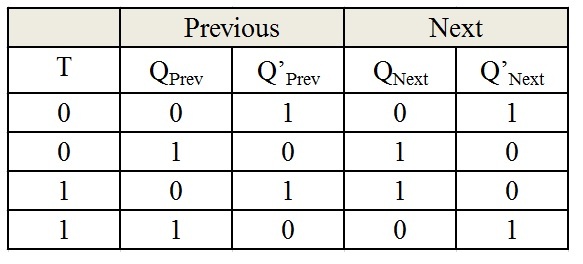
**D FF Truth Table**

|  |  |
| --- | --- |
| **D** | **O/P** |
| 0 | 0 |
| 1 | 1 |

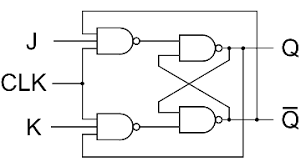


**TFF Truth Table**





**Diagram of JK Flip Flop using NAND gates**



**Conclusion:**

Hence design of JK Flip flop, D flip flop, T flip flop using NAND Gates & verification of the same flip flop using IC7476 is implemented.

**Post Lab Descriptive Questions**

1. How does a JK flip-flop differ from an SR flip-flop in its basic operation?
2. What is use of characteristic and excitation table?
3. How many flip flops due you require storing the data 1101?
4. Describe the basic difference between pulse-triggered and edge-triggered flip-flops.

.

1 JK flip flop is made up of SR flip flop. On high inputs JK toggles the previous values whereas SR enters “*Race*” state. When S is low the output is “*Set”* and when R is low the output is cleared that is “0”. In JK Flip Flop the output is according to J input except for the cases when both inputs are same. In JK Flip Flop there is a clock, preset and clear option as well. No such thing exists in SR flip flop.

2 In [electronics design](https://en.wikipedia.org/wiki/Electronics_design), an **excitation table** shows the minimum inputs that are necessary to generate a particular next state (in other words, to **"excite"** it to the next state) when the current state is known. They are similar to [truth tables](https://en.wikipedia.org/wiki/Truth_table) and [state tables](https://en.wikipedia.org/wiki/State_table), but rearrange the data so that the current state and next state are next to each other on the left-hand side of the table, and the inputs needed to make that state change happen are shown on the right side of the table. In order to obtain the excitation table of a [flip-flop](https://en.wikipedia.org/wiki/Flip-flop_(electronics)), one needs to draw the Q(t) and Q(t+1) for all possible cases (e.g., 00,01,10 and 11), and then make the value of flip-flop such that on giving this value, one shall receive the input as Q(t+1) as desired.

3 Since the number is 4 bits long we will require 4 JK flip flops to store 1101.

4 A pulse can be generated by moving a switch from Low to High then back to Low. So the pulse-triggered FF needs a complete pulse Low -High - Low. And edge triggered can be divided into two:  
Positive edge triggered ---> Low to High transition.  
Negative edge triggered ---> High to Low transition.

**Experiment / Assignment / Tutorial No. \_\_\_\_\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Batch: Roll No.: Experiment / assignment / tutorial No.: 6** |

|  |
| --- |
| **Title:** 3-bit Asynchronous Counter |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** Design of 3 bit asynchronous counter using JK flip flop

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO2:** Use different minimization technique and solve combinational circuits, synchronous & asynchronous sequential circuits.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

* R. P. Jain, “Modern Digital Electronics”, Tata McGraw Hill
* M .Morris Mano, “Digital Logic & computer Design”, PHI
* A.P.Godse, D.A.Godse, “Digital Logic Design”
* http://www.fatih.edu.tr/~aliadam/EEE122A/EEE122Ch6COUNTERS.pdf

**Pre Lab/ Prior Concepts:**

A counter is a register capable of counting number of clock pulse arriving at its clock input. Counter represents the number of clock pulses arrived. A specified sequence of states appears as counter output. This is the main difference between a register and a counter. There are two types of counter, synchronous and asynchronous. In synchronous common clock is given to all flip flop and in asynchronous first flip flop is clocked by external pulse and then each successive flip flop is clocked by Q or Q output of previous stage. A soon the clock of second stage is triggered by output of first stage. Because of inherent propagation delay time all flip flops are not activated at same time which results in asynchronous operation.

**Implementation Details:**

**Truth Table for 3 bit UP counter**

**Logic Diagram for 3 bit UP counter (Negative edge)**

**Timing Diagram for 3 bit UP counter (Negative edge)**

**Truth Table for 3 bit DOWN counter**

**Logic Diagram for 3 bit DOWN counter (Negative edge)**

**Timing Diagram for 3 bit DOWN counter**

**Conclusion:**

**Post Lab Descriptive Questions**

1. Draw logic diagram for mod – 6 asynchronous up counter.

**Experiment / Assignment / Tutorial No. \_\_\_\_\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Batch: Roll No.: Experiment / assignment / tutorial No.: 7** |

|  |
| --- |
| **Title:** 3-bit Synchronous Counter |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** Design of 3 bit Synchronous counter using JK flip flop

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO2:** Use different minimization technique and solve combinational circuits, synchronous & asynchronous sequential circuits.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

* R. P. Jain, “Modern Digital Electronics”, Tata McGraw Hill
* M .Morris Mano, “Digital Logic & computer Design”, PHI
* A.P.Godse, D.A.Godse, “Digital Logic Design”
* http://www.fatih.edu.tr/~aliadam/EEE122A/EEE122Ch6COUNTERS.pdf

**Pre Lab/ Prior Concepts:**

A counter is a register capable of counting number of clock pulse arriving at its clock input. Counter represents the number of clock pulses arrived. A specified sequence of states appears as counter output. This is the main difference between a register and a counter. There are two types of counter, synchronous and asynchronous. In synchronous common clock is given to all flip flop and in asynchronous first flip flop is clocked by external pulse and then each successive flip flop is clocked by Q or Q output of previous stage. A soon the clock of second stage is triggered by output of first stage. Because of inherent propagation delay time all flip flops are not activated at same time which results in asynchronous operation.

**Implementation Details:**

**Characteristic Table for 3 bit UP counter**

|  |  |  |  |
| --- | --- | --- | --- |
| **Q** | **Qt+1** | **J** | **K** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Truth Table for 3 bit UP Counter**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Present State**  **QA QB QC** | **Next State**  **QA+1 Q B+1 QC+1** | **A**  **JA KA** | **B**  **JB KB** | **C**  **JC KC** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**K Map**

**Logic Diagram for 3 bit UP counter**

**Timing Diagram for 3 bit UP counter**

**Conclusion:**

**Post Lab Descriptive Questions**

1. Draw logic diagram for mod-2 synchronous down counter.

**Experiment / Assignment / Tutorial No. \_\_\_\_\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Batch: Roll No.: Experiment / assignment / tutorial No.: 8** |

|  |
| --- |
| **Title:**  Shift Register |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** To implement the SISO, SIPO, PISO, PIPO shift register using D flips flop

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO2:** Use different minimization technique and solve combinational circuits, synchronous & asynchronous sequential circuits.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

* R. P. Jain, “Modern Digital Electronics”, Tata McGraw Hill
* M .Morris Mano, “Digital Logic & computer Design”, PHI
* A.P.Godse, D.A.Godse, “Digital Logic Design”

**Pre Lab/ Prior Concepts:**

A register is capable of shifting its binary information in one or both directions is known as shift register. The logical configuration of shift register consist of a D-Flip flop cascaded with output of one flip flop connected to input of next flip flop. All flip flops receive common clock pulses which causes the shift in the output of the flip flop.The simplest possible shift register is one that uses only flip flop. The output of a given flip flop is connected to the input of next flip flop of the register. Each clock pulse shifts the content of register one bit position to right.

The basic types of shift registers are

* Serial In - Serial Out
* Serial In - Parallel Out
* Parallel In - Serial Out
* Parallel In - Parallel Out
* Bidirectional shift registers.

**Implementation Details:**

**Logic Diagram**

**Serial in Serial Out**

**Truth table**

**Serial In - Parallel Out**

**Truth table**

**Parallel In Serial Out**

**Truth table**

**Parallel In Parallel Out**

**Truth table**

**Conclusion:**

**Post Lab Descriptive Questions**

1. Draw logic diagram for universal shift register using 4:1 MUX.
2. Develop the logic diagram for the shift register using JK flip-flop to replace the D flip flop?
3. How many clock pulses are required to enter a byte of data serially into an 8-bit shift register?

**Experiment / Assignment / Tutorial No. \_\_\_\_\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Batch: Roll No.: Experiment / assignment / tutorial No.: 9** |

|  |
| --- |
| **Title:** VHDL programming for gates |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** Implements a simple OR, AND, XOR, NOR, NAND, XNOR gate in VHDL

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO4:** Implement digital networks using VHDL.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

* J. Bhasker, “VHDL Primer”, Pearson Education
* Douglas L. Perry, “VHDL Programming by Example”, Tata McGraw Hill
* http://esd.cs.ucr.edu/labs/tutorial/

**Pre Lab/ Prior Concepts:**

VHDL is an acronym for VHSlC Hardware Description Language (VHSIC is an acronym for Very High Speed Integrated Circuits). It is a hardware description language that can be used to model a digital system at many levels of abstraction ranging from the algorithmic level to the gate level. The complexity of the digital system being modeled could vary from that of a simple gate to a complete digital electronic system, or anything in between. The digital system can also be described hierarchically. Timing can also be explicitly modeled in the same description.

**VHDL Programming Structure**

Entity and Architecture are the two main basic programming structures in VHDL.

**Entity:** Entity can be seen as the black box view of the system. We define the inputs and outputs of the system which we need to interface. It is used to declare the I/O ports of the circuit.

Eg:

Entity ANDGATE is

Port (A: in std\_logic;

B: in std\_logic;

Y: out std\_logic);

End entity ANDGATE;

Entity name ANDGATE is given by the programmer, each entity must have a name.

**Architecture:** Architecture defines what is in our black box that we described using ENTITY. The description code resides within architecture portion. Either behavioral or structural models can be used to describe our system in the architecture. In Architecture we will have interconnections, processes, components, etc.

Eg:

Architecture AND1 of ANDGATE is

--declarations

Begin

--statements

Y <= A AND B;

End architecture AND1;

Entity name or architecture name is user defined. Identifiers can have uppercase alphabets, lowercase alphabets, and numbers and underscore (\_). First letter of identifier must be an alphabet and identifier cannot end with an underscore. In VHDL, keywords and user identifiers are case insensitive.

VHDL is strongly typed language i.e. every object must be declared. Standardized design libraries are typically used and are included prior to the entity declaration. This is accomplished by including the code "library ieee;" and "use ieee.std\_logic\_1164.all;"

**Implementation Details:**

**VHDL program code**

**OR**

**AND**

**XOR**

**NOT**

**NOR**

**NAND**

**XNOR**

**Conclusion:**

**Post Lab Descriptive Questions**

1. What are two types of HDL?

**Experiment / Assignment / Tutorial No. \_\_\_\_\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Batch: Roll No.: Experiment / assignment / tutorial No.:10** |

|  |
| --- |
| **Title:** VHDL programming for multiplexer |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** Implements a simple Multiplexer in VHDL

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO4:** Implement digital networks using VHDL.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

* J. Bhasker, “VHDL Primer”, Pearson Education
* Douglas L. Perry, “VHDL Programming by Example”, Tata McGraw Hill
* http://esd.cs.ucr.edu/labs/tutorial/

**Pre Lab/ Prior Concepts:**

VHDL is an acronym for VHSlC Hardware Description Language (VHSIC is an acronym for Very High Speed Integrated Circuits). It is a hardware description language that can be used to model a digital system at many levels of abstraction ranging from the algorithmic level to the gate level. The complexity of the digital system being modeled could vary from that of a simple gate to a complete digital electronic system, or anything in between. The digital system can also be described hierarchically. Timing can also be explicitly modeled in the same description.

**VHDL Programming Structure**

Entity and Architecture are the two main basic programming structures in VHDL.

**Entity:** Entity can be seen as the black box view of the system. We define the inputs and outputs of the system which we need to interface. It is used to declare the I/O ports of the circuit.

Eg:

Entity ANDGATE is

Port (A: in std\_logic;

B: in std\_logic;

Y: out std\_logic);

End entity ANDGATE;

Entity name ANDGATE is given by the programmer, each entity must have a name.

**Architecture:** Architecture defines what is in our black box that we described using ENTITY. The description code resides within architecture portion. Either behavioral or structural models can be used to describe our system in the architecture. In Architecture we will have interconnections, processes, components, etc.

Eg:

Architecture AND1 of ANDGATE is

--declarations

Begin

--statements

Y <= A AND B;

End architecture AND1;

Entity name or architecture name is user defined. Identifiers can have uppercase alphabets, lowercase alphabets, and numbers and underscore (\_). First letter of identifier must be an alphabet and identifier cannot end with an underscore. In VHDL, keywords and user identifiers are case insensitive.

VHDL is strongly typed language i.e. every object must be declared. Standardized design libraries are typically used and are included prior to the entity declaration. This is accomplished by including the code "library ieee;" and "use ieee.std\_logic\_1164.all;"

**Implementation Details:**

**VHDL simulation output**

**Conclusion:**

**Post Lab Descriptive Questions**

1. Write VHDL program for 2:1 multiplexer.