**DLD LAB EXPERIMENT**



**Name: Basit Ali**

**Subject: - Digital Logic and Design**

**Submitted to: - Sadaqat Hussain**

**Registration No: - S23BSCS012**

**Section: - A**

**Department: - Computer Science**

**Table of Contents:**

**Experiment 1: ...........................................................................................................4**

Logic Gates (AND, OR and NOT) ... ……………….…...….…**Date: 6 and 7/ SEP/ 2023**

NOT Gate...................................................................................................................4

OR Gate.....................................................................................................................5

AND Gate...................................................................................................................6

**Experiment 2: ...........................................................................................................7**

Logic Gates (NOR and NAND Gates) ……………………. **Date:** 13 and 14**/** SEP**/** 2023

NOR Gate..................................................................................................................7

NAND Gate................................................................................................................8

**Experiment 3: ..........................................................................................................9**

Logic Gates (XOR and XNOR Gates) …………………… **Date:** 20 and 21**/** SEP**/** 2023

XOR Gate.................................................................................................................9

XNOR Gate..............................................................................................................10

**Experiment 4: ........................................................................................................11**

Half Adder …………………………………………………….......... **Date:** 27**/** SEP**/** 2023

Purpose....................................................................................................................11

Diagram…………………………………………………………………………...............12

**Experiment 5: .........................................................................................................12**

Full Adder …………………………………………………….......... **Date:** 28**/** SEP**/** 2023

Purpose....................................................................................................................12

Diagram…………………………………………………………………………...............13

**Experiment 6: .........................................................................................................13**

DE Morgan Laws……….…………………………………...**Date:** 11 and 12 **/** OCT**/** 2023

1st Law.................................................................................................................13,14

2nd Law…………………………………………………………………………............14,15

**Experiment 7: ........................................................................................................ 15**

Multiplexer (4 to 1) …………………………………………. **Date:** 18 and 19 **/** OCT**/** 2023

4 X 1 Multiplexer..............................................................................................……...15

4 X 1 Circuit Diagram................................................................................................16

**Experiment 8: ..........................................................................................................16**

DE multiplexer (4 to 1) ………………….………………….. **Date:** 25 and 26 **/** OCT**/** 2023

4 X 1 Demultiplexer...................................................................................................16

4 X 1 Circuit Diagram................................................................................................17

**Experiment 9: ......................................................................................................... 17**

Latch………………………………………………………….…**Date:** 1 and 2 **/** NOV**/** 2023

SR Latch Using NAND Gate.................................................................................17,18

SR Latch Using NOR Gate...................................................................................18,19

**Experiment 10: ....................................................................................................... 20**

S-R Flip Flop………………………………………………………... **Date:** 8 **/** NOV**/** 2023

S-R Flip Flop Using NAND Gate..........................................................................20,21

**Experiment 11: ....................................................................................................... 21**

J-K Flip Flop………………………………………………………… **Date:** 15 **/** NOV**/** 2023

J-K Flip Flop Using NAND Gate......................................................................21,22,23

**Experiment 12: ........................................................................................................23**

D and T Flip Flop…………………………………………………... **Date:** 16 **/** NOV**/** 2023

D Flip Flop Using NAND Gate..............................................................................23,24

T Flip Flop Using NAND and NOR Gate...............................................................25,26

**Experiment 1:**

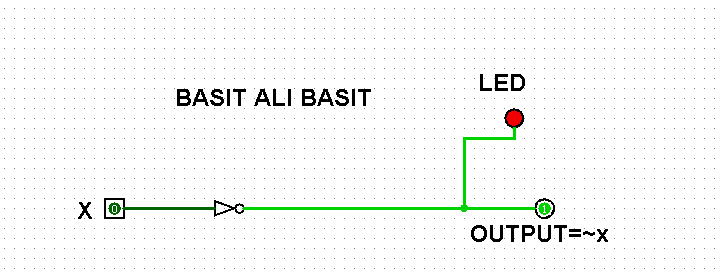
**Logic Gates (NOT, OR and AND Gates)**

**1. NOT GATE:**

**Primary purpose:**

The primary purpose of a NOT gate, also known as an inverter, is to perform the logical NOT operation. The NOT gate takes a single input signal and produces an output signal that is the opposite or complement of the input. In other words, if the input is HIGH (1), the output is LOW (0), and vice versa.

**Symbol:**

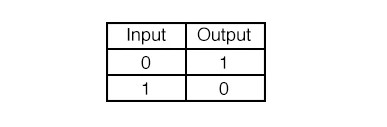
****

The NOT gate is a fundamental building block in digital circuit design and is used in various applications, including:

1. Inverting the logic level of a signal.
2. Combining with other gates to perform more complex logical operations.
3. In certain configurations, NOT gates are used in the design of memory elements and flip-flops.
4. Inverting control signals in digital systems.

NOT gates play a crucial role in the construction of digital circuits and are essential for designing a wide range of digital devices and systems.

**Truth Table:**



**Working:**

NOT Gate invert the Input means there is two possibilities:

1. if we give input 0 then it gives output 1.
2. if we give input 1 then it gives output 0.

**2. OR Gate:**

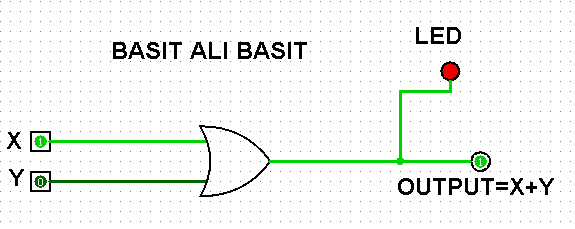
**Primary purpose:**

The primary purpose of an OR gate in digital electronics is to perform the logical OR operation. An OR gate takes two binary inputs (0 or 1) and produces a single binary output based on the following logic:

If at least one of the inputs is 1, the output is 1.  If both inputs are 0, the output is 0.

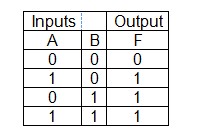
The OR gate is often represented by the symbol "" or by the word "OR" and is another fundamental building block in digital circuits. It is used to combine multiple signals or conditions in a way that allows flexibility and decision-making in electronic systems.

**Symbol:**



OR gates are widely used in the design of logical circuits, providing a means to create conditions where an output is true if one or more specific input conditions are met. They are essential components in the implementation of Boolean algebra and are used in various electronic devices and systems, including digital computers, calculators, and other digital systems.

**Truth Table:**



**Working:**

OR Gate give the output high when at least one is input is high there is four possibilities:

1. if we give input 0 and 0 then it gives output 0. 2. if we give input 0 and 1 then it gives output 1. 3. if we give input 1 and 0 then it gives output 1.

4. if we give input 1 and 1 then it gives output 1.

**3. AND Gate:**

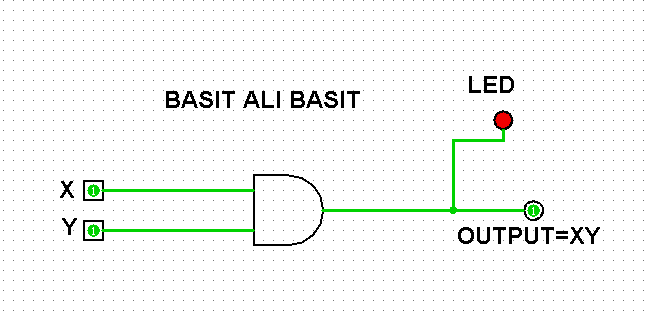
**Primary purpose:**

The primary purpose of an AND gate in digital logic is to perform a logical AND operation on its input signals.\ The AND gate has two or more inputs and produces a single output based on the following logic:

* If both inputs are 1, the output is 1.
* If any of the inputs is 0, the output is 0.

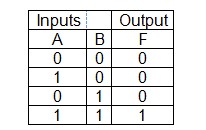
The AND gate is often represented by the symbol "&" or by the word "AND" and is a fundamental building block in digital circuits. It is used in various electronic devices and systems to control the flow of information or signals based on specific logical conditions.

**Symbol:**

****

AND gates are crucial in the design of logical circuits, such as arithmetic units, memory units, and control units in computers and other digital systems. They are used to implement logical operations and conditions that are fundamental to the functioning of digital electronics.

**Truth Table:**



**Working:**

AND Gate give the output low when at least one input is low or it give high when all input is high there are four possibilities:

1. if we give input 0 and 0 then it gives output 0. 2. if we give input 0 and 1 then it gives output 0. 3. if we give input 1 and 0 then it gives output 0.

4. if we give input 1 and 1 then it gives output 1.

**Experiment 2:**

**Logic Gates (NOR and NAND Gates)**

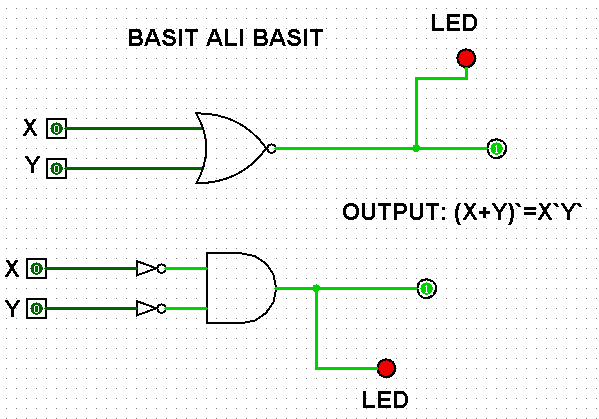
**1. NOR Gate:**

**Primary purpose:**

The NOR gate in digital electronics is a logical gate that performs the logical NOR (NOT-OR) operation. It has two binary inputs and one binary output. The primary purpose of a NOR gate is to produce an output of 1 (or HIGH) only when both of its inputs are 0 (or LOW).

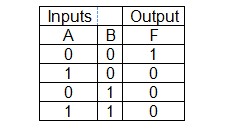
NOR gates are often represented by the symbol "" or by the word "NOR." They are particularly useful in digital circuit design for creating certain logical functions and conditions. NOR gates are considered universal gates, meaning that any other logic gate (AND, OR, NOT) can be constructed using only NOR gates.

**Symbol:**



NOR gates find applications in various electronic systems, including digital computers, memory circuits, and control units, where their ability to generate specific logical conditions proves valuable in the overall functioning of the system.

**Truth Table:**



**Working:**

NOR Gate give the output high when all the inputs are low or it give low when at least one input is high there are four possibilities:

1. if we give input 0 and 0 then it gives output 1.
2. if we give input 0 and 1 then it gives output 0.
3. if we give input 1 and 0 then it gives output 0.
4. if we give input 1 and 1 then it gives output 0.

**2-NAND Gate:**

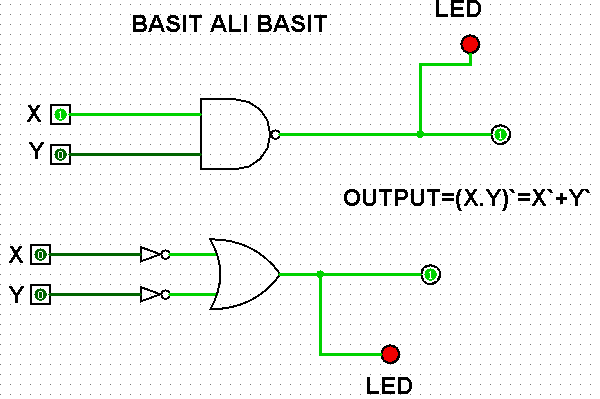
**Primary purpose:**

The NAND gate in digital electronics is a logical gate that performs the logical NAND (NOT-AND) operation. It has two binary inputs and one binary output. The primary purpose of a NAND gate is to produce an output of 0 (or LOW) only when both of its inputs are 1 (or HIGH).

The output is 1 if at least one of the inputs is 0.

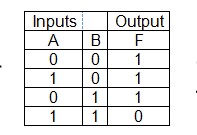
NAND gates are often represented by the symbol "" or by the word "NAND." They are also considered universal gates, meaning that any other logic gate (AND, OR, NOT) can be constructed using only NAND gates.

**Symbol:**



NAND gates are widely used in digital circuit design and play a fundamental role in constructing various logical functions. They are crucial components in electronic systems, including digital computers, memory circuits, and control units, where their ability to generate specific logical conditions is essential for the proper functioning of the system.

**Truth Table:**



**Working:**

NAND Gate give the output high when at least one input is low or it give low when all input is high there are four possibilities:

1. if we give input 0 and 0 then it gives output 1. 2. if we give input 0 and 1 then it gives output 1. 3. if we give input 1 and 0 then it gives output 1.

4. if we give input 1 and 1 then it gives output 0.

**Experiment 3:**

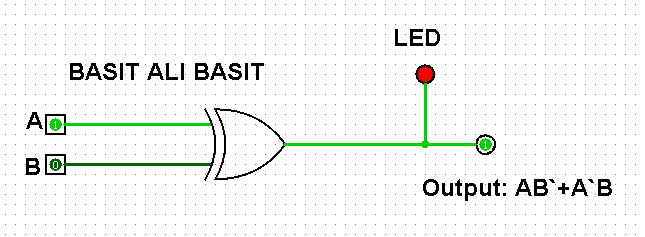
**Logic Gates (XOR and XNOR Gates)**

**1. XOR Gate:**

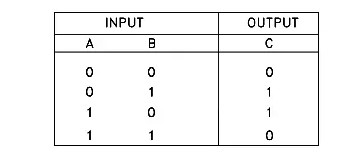
**Primary purpose:**

1. The XOR gate outputs 1 when the number of true inputs is odd.
2. Conversely, if both inputs are the same (both true or both false), the output is 0.
3. XOR is often used for error checking and parity generation in communication systems.
4. It's a key component in adder circuits, contributing to binary addition in computers.
5. XOR operations are fundamental in cryptography, where they are used for encryption.
6. XOR gates are essential for creating flip-flops and sequential logic circuits.
7. XOR's unique behavior makes it valuable in various applications, offering versatile functionality in digital circuitry.

**Symbol:**

****

**Truth Table:**



**Working:**

* A and B are the inputs.
* The AND gate produces an output when both A and B are 1.
* The OR gate produces an output when at least one of A or B is 1. The NOT gate inverts the A and B before AND gate.
* The XOR gate produces the final output by combining the outputs of the OR gate and the inverted AND gate.

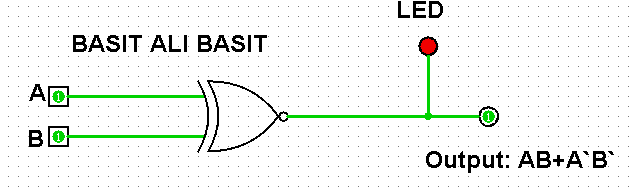
This arrangement of gates creates the XOR (exclusive OR) functionality, where the output is 1 when the inputs are different and 0 when the inputs are the same.

**2.XNOR Gate:**

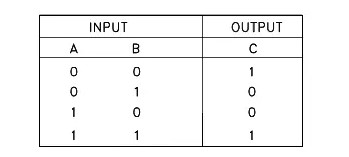
**Primary purpose:**

1. The XNOR (exclusive NOR) gate in digital electronics primarily compares two binary inputs.
2. It produces a 1 output when both inputs are the same (both true or both false).
3. Conversely, it outputs 0 when the inputs are different (one is true, and the other is false).
4. XNOR gates are often used for equality checking in digital circuits.
5. They play a key role in creating latch and flip-flop circuits, crucial for memory storage in computers.
6. XNOR operations are employed in certain arithmetic circuits, such as binary comparators.
7. Their primary function is to determine equivalence, making them valuable in various logical and arithmetic applications.

**Symbol:**

****

**Truth Table:**



**Work:**

* A and B are the inputs.
* The first AND gate produces an output when both A and B are 1.
* The OR gate produces an output when either the output of the first AND gate is 1.
* The NOT gate inverts the Input.
* The second AND gate will invert the both input.

This arrangement of gates creates the XNOR (exclusive NOR) functionality, where the output is 1 when the inputs are equal and 0 when the inputs are different.

**Experiment 4:**

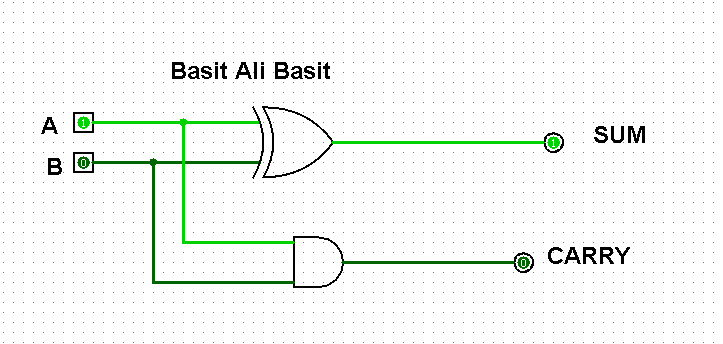
**Half Adder:**

**Purpose:**

A half adder serves as a basic digital circuit with the following primary purposes:

* Adds two binary digits, denoted as A and B.
* Generates the sum (S) bit, representing the result of binary addition using XOR logic.
* Produces the carry (C) bit, indicating if there is a carry-over from the addition using AND logic.
* Functions as a foundational component for constructing more complex arithmetic circuits, with full adders used for multi-bit addition.

**Circuit Diagram:**

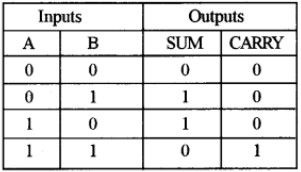


**Procedure:**

Creating a half adder circuit involves following step:

* Gather AND gates, NOT gates, input switches (for A and B), LEDs (for Sum and Carry outputs).
* Connect A and B switches to separate AND gates. Connect the output of each AND gate to a NOT gate. Connect the outputs of the NOT gates to a final AND gate. The output of this final AND gate represents the sum (S).
* Connect A and B switches directly to an AND gate. The output of this AND gate represents the carry (C).

**Truth Table:**



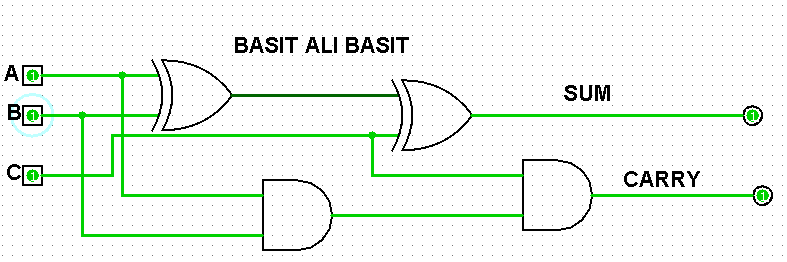
**Experiment 5**

**Full Adder:**

**Purpose:**

* The primary purpose of a full adder is to add three binary digits - two single-bit inputs (A and B) and a carry input (C). It produces two outputs: the sum (S) and a carry output. This allows it to perform the addition of more significant bits in multi-bit binary numbers.
* Full adders are often cascaded together to perform addition on multi-bit binary numbers. The carry output from one full adder serves as the carry input to the next, enabling the addition of numbers with more than one bit.
* Full adders are fundamental building blocks for constructing various arithmetic circuits, such as binary adders and subtractors. They are essential components in the design of central processing units (CPUs) and other complex digital systems where arithmetic operations are required.
* Full adders contribute to the versatility of binary arithmetic operations. They can handle the addition of three bits in a single unit, making them more efficient than using multiple half adders for the same purpose. Their ability to propagate carry information is crucial for accurate and efficient binary arithmetic.

**Circuit Diagram:**



**Procedure:**

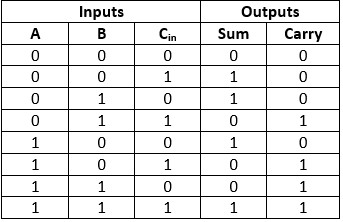
Creating a full adder using XOR and NOT gates involves combining these gates to achieve the functionality of adding three binary digits (A, B, and a carry input, C). Here's a simplified step-by-step procedure in four points:

* Gather XOR gates, AND gates, NOT gates, input switches (for A, B, and C), LEDs (for Sum and Carry outputs), a breadboard, and jumper wires.

* Connect A and B switches to separate XOR gates. Connect the outputs of the XOR gates to another XOR gate along with the C input. This represents the sum (S) output.

* Connect A and B switches to separate AND gates. Connect the output of the AND gate corresponding to A to one input of an OR gate. Connect the output of the AND gate corresponding to B to another input of the OR gate. Connect the output of the XOR gate (from the S wiring) to a NOT gate and then connect its output to the third input of the OR gate. This represents the carry (C) output.

**Truth Table:**



**Experiment 6:**

**DE Morgan's** **Laws:**

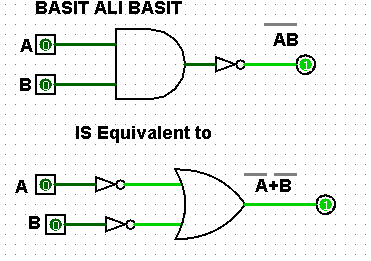
* **First law:**

**Purpose:**

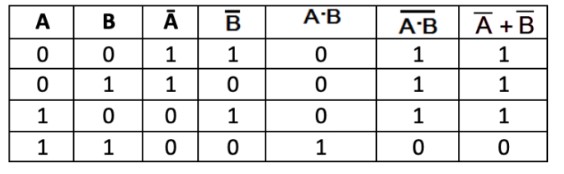
De Morgan's First Law is a fundamental principle in Boolean algebra, describing the relationship between the complement (NOT) of a logical AND operation and the logical OR operation. Here are three key purposes of De Morgan's First Law:

* + De Morgan's First Law provides a method to simplify complex logical expressions by allowing the transformation of a complemented AND operation into an OR operation with complemented terms. This simplification is valuable in the design and analysis of digital circuits and Boolean algebraic expressions.
  + The law offers an alternative way to express logical relationships, allowing designers and analysts to choose the form that is most convenient or intuitive for a given problem. It facilitates flexibility in representing and manipulating logical expressions.
  + De Morgan's First Law is often applied in the design and implementation of digital circuits. By understanding and utilizing this law, engineers can optimize the arrangement of gates in a circuit, potentially reducing the number of gates needed and improving efficiency in terms of both hardware and computational resources.

**Circuit Diagram:**

****

**Truth Table:**



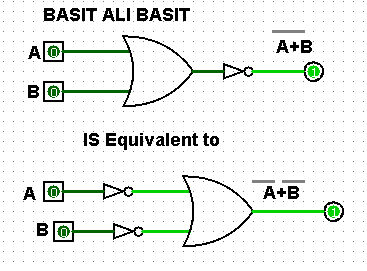
* **Second Law:**

**Purpose:**

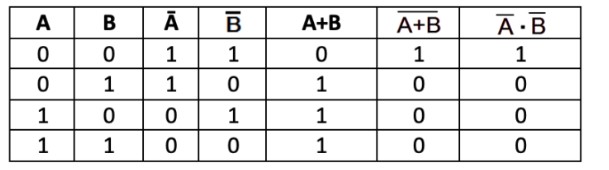
De Morgan's Second Law is another fundamental principle in Boolean algebra, providing a relationship between the complement (NOT) of a logical OR operation and the logical AND operation. Here are three key purposes of De Morgan's Second Law:

* + Similar to the first law, De Morgan's Second Law enables the simplification of complex logical expressions. It allows the transformation of a complemented OR operation into an AND operation with complemented terms. This simplification is valuable in Boolean algebraic manipulation, making expressions more manageable and aiding in circuit design.
  + The law offers an alternative representation for logical relationships involving complements and OR operations. This flexibility allows for different ways of expressing the same logic, providing a useful tool for designers and analysts to choose the most convenient form for a given problem or context.
  + De Morgan's Second Law is applied in the design and implementation of digital circuits, similar to the first law. Engineers use this law to optimize the arrangement of gates, potentially reducing the complexity of circuits and improving efficiency. Understanding and applying both De Morgan's laws are crucial for working with Boolean algebra and designing digital systems.

**Circuit Diagram:**

****

**Truth Table:**



**Experiment 7:**

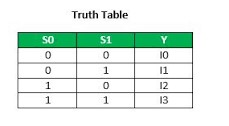
**Multiplexer (4 to 1):**

**Inputs and Outputs:**

Identify the inputs (I0, I1, I2, I3) selection lines (S0, S1) and the output (Y) of the 4-to-1 multiplexer.

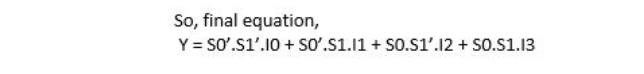
**Truth Table:**

Create a truth table showing the output (Y) for all combinations of data inputs and selection inputs (S 0, S 1).



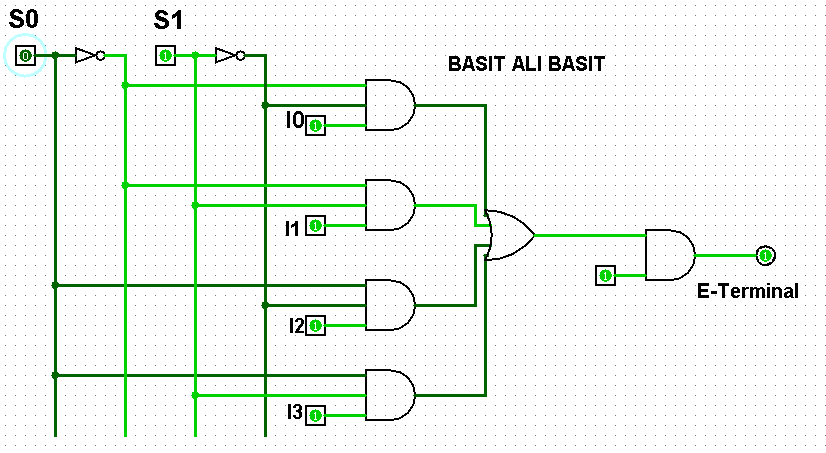
**Expression in SOP Form:**

Express the output (Y) in Sum of Products (SOP) form using the truth table.

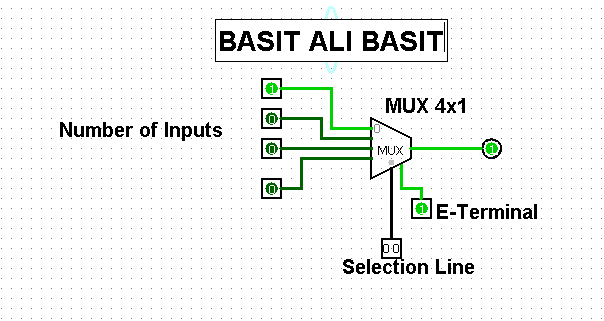


**Circuit Implementation:**

Implement the 4-to-1 MUX circuit using NAND gates based on the SOP expression, using De Morgan's theorem for conversion if necessary.



**Diagram:**



**Experiment 8:**

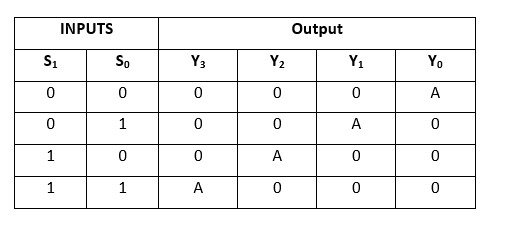
**DE multiplexer (4 to 1):**

**Identify Inputs and Outputs:**

Identify the inputs (S0 S1) and outputs (Y0, Y1, Y2, Y3) of the DE multiplexer (Demuxer).

**Truth Table:**

Create a truth table to represent the output states for all combinations of inputs.



**Boolean Expression:**

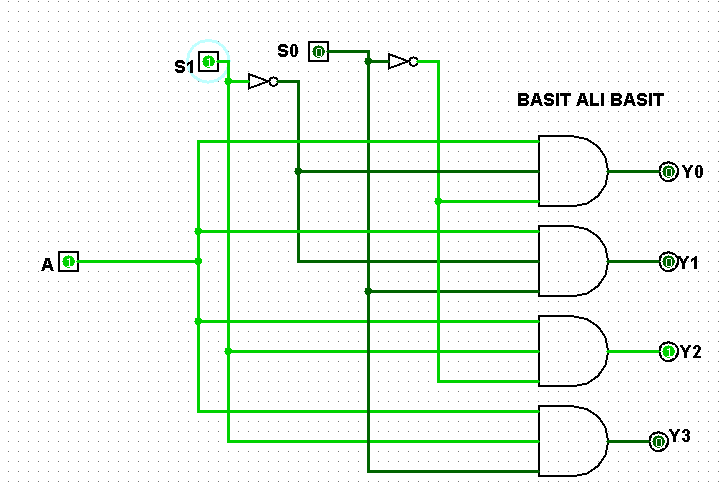
Formulate the Boolean expression for the DE multiplexer based on the truth table.

The logical expression of the term Y is as follows:

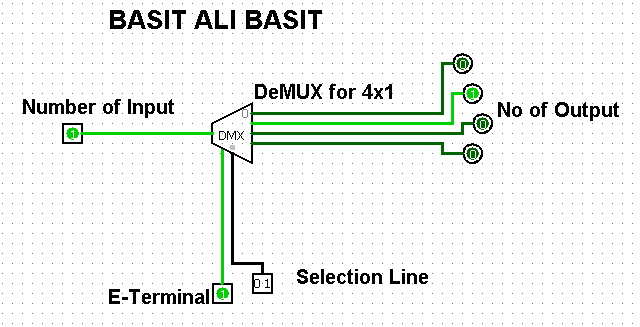
Y0=S1'S0'A y1=S1'S0 A y2=S1 S0'A y3=S1 S0 A

**Circuit Implementation:**

Implement the DE multiplexer circuit using logic gates, considering the Boolean expression and the identified inputs and outputs.



**Diagram:**

****

**Experiment 9:**

**S-R Latch:**

**A. SR Latch Using NAND Gate:**

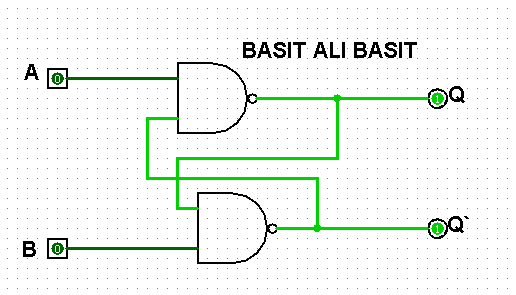
**Purpose:**

The SR latch, constructed using NAND gates, serves several important purposes in digital circuit design. Here are three key points regarding the purpose of an SR latch using NAND gates:

* + - * An SR latch is a fundamental memory element in digital circuits. It can store and remember a binary state (0 or 1) until explicitly changed. The NAND-based SR latch achieves this memory function by utilizing feedback loops within the circuit.
      * The NAND-based SR latch operates as a bi stable latch, meaning it has two stable states. It can be set to one state (Q = 1, Q' = 0) by activating the Set input (S) and reset to the other state (Q = 0, Q' = 1) by activating the Reset input (R). When both Set and Reset inputs are inactive, the latch maintains its current state, allowing it to function as a basic memory storage element.
      * The NAND-based SR latch allows controlled storage and transfer of information. By manipulating the Set and Reset inputs based on external signals or conditions, designers can control when and how data is stored or transferred within a digital system. This controlled behavior is essential for sequential logic circuits, where the order of operations is critical.

**Circuit Diagrams:**

**S-R Latch Using NAND Gates:**



**Truth Table:**

**Truth Table of S-R Latch Using NAND Gates:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S** | **R** | **Q** | **Q`** | **State** |
| **0** | **0** | **X** | **X** | **Invalid State** |
| **0** | **1** | **0** | **1** | **Reset** |
| **1** | **0** | **1** | **0** | **Set** |
| **1** | **1** | **Q** | **Q`** | **Hold** |

**Procedure:**

**Procedure of S-R Latch Using NAND Gates:**

Procedure to make a S-R latch using NAND gate in logism software is below:

* + - Take a two NAND gates from left corner and connect them with inputs A and B.
    - Pass the inputs through the gates and after the output of first gate will be input for second gate and also output of second gate will become input for first gate.  Store output in Q and next output in Q`. **B. SR Latch Using NOR Gate:**

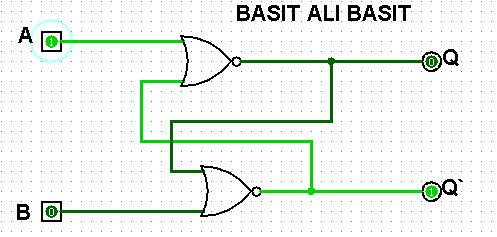
**Purpose:**

An SR latch constructed using NOR gates serves several purposes in digital circuit design. Here are three key points regarding the purpose of an SR latch using NOR gates:

* + - An SR latch is a fundamental memory element in digital circuits. It can store and remember a binary state (0 or 1) until explicitly changed. The NOR-based SR latch achieves this memory function by utilizing feedback loops within the circuit, similar to the NAND gate version.
    - The NOR-based SR latch operates as a bi stable latch, meaning it has two stable states. It can be set to one state (Q = 1, Q' = 0) by activating the Set input (S) and reset to the other state (Q = 0, Q' = 1) by activating the Reset input (R). When both Set and Reset inputs are inactive, the latch maintains its current state, allowing it to function as a basic memory storage element.
    - The NOR-based SR latch allows controlled storage and transfer of information. By manipulating the Set and Reset inputs based on external signals or conditions, designers can control when and how data is stored or transferred within a digital system. This controlled behavior is essential for sequential logic circuits, where the order of operations is critical.

**Circuit Diagram:**

**S-R Latch Using NOR Gates:**

****

**Truth Tables:**

**Truth Table of S-R Latch Using NOR Gates:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S** | **R** | **Q** | **Q`** | **State** |
| **0** | **0** | **Q** | **Q`** | **Hold** |
| **0** | **1** | **0** | **1** | **Reset** |
| **1** | **0** | **1** | **0** | **Set** |
| **1** | **1** | **X** | **X** | **Invalid State** |

**Procedure of S-R Latch Using NOR Gates:**

Procedure to make a S-R latch using NOR gate in logism software is below:

* + Take a two NOR gates from left corner and connect them with inputs A and B.
  + Pass the inputs through the gates and after the output of first gate will be input for second gate and also output of second gate will become input for first gate.
  + Store output in Q and next output in Q.

**Experiment 10:**

**S-R Flip Flops:**

**a. S-R Flip Flop Using NAND Gates:**

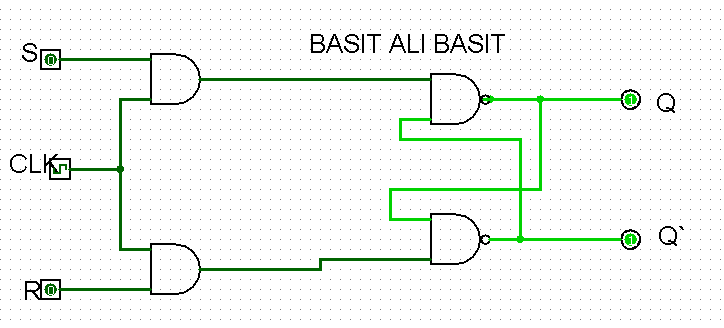
**Purpose:**

* A Set-Reset (SR) flip-flop, also known as a Set-Reset latch or simply an SR latch, is a fundamental digital electronic circuit element. It is commonly constructed using NAND gates. The purpose of an SR flip-flop is to store one bit of information. It has two inputs, S (Set) and R (Reset), and two outputs, Q and Q' (complement of Q).
* The NAND gate implementation is popular due to its simplicity.

The basic SR flip-flop using NAND gates has two cross-coupled NAND gates, and it is often called an SR NAND latch. The circuit's simplicity allows for easy understanding and implementation.

* However, it's important to note that the SR flip-flop has a potential issue called the "set-reset" or "race" condition. If both S and R inputs are active at the same time, the output can oscillate rapidly between 0 and 1, leading to an unstable state. To avoid this, more complex flip-flop designs, such as the D flip-flop or JK flip-flop, are often used in practical applications. These designs include additional logic to prevent the simultaneous activation of Set and Reset inputs.

**Diagram:**



**Working:**

Working of SR Flip flop:

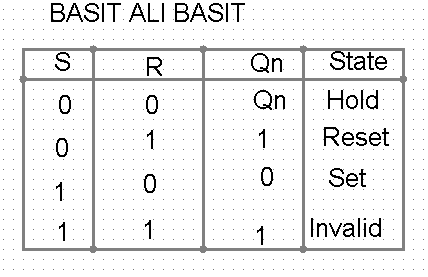
* When the Set input (S) is activated (high or 1), it forces the Q output to go high (1). At the same time, the complement output Q' goes low (0). This sets the flip-flop to the state where Q is 1.
* When the Reset input (R) is activated (high or 1), it forces the Q output to go low (0). Simultaneously, the complement output Q' goes high (1). This resets the flip-flop to the state where Q is 0.
* If both S and R inputs are activated simultaneously, it can lead to an unpredictable state. In practice, this condition should be avoided in order to maintain a reliable operation of the flip-flop.

**Procedure:**

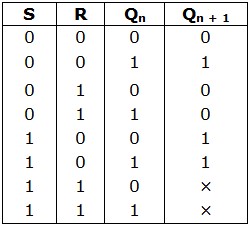
Creating a S-R flip flop using NAND gates involves following steps:

* Take a two NAND Gates which take two inputs name as S, R and Clock (CLK).
* Clock will always 1.
* Take two NAND gates more, than the previous output will give as input and the next output will give as inputs  Q and Q` will be the final Outputs.

**Truth Table:**

****

**Characteristics Table:**



**Experiment 11:**

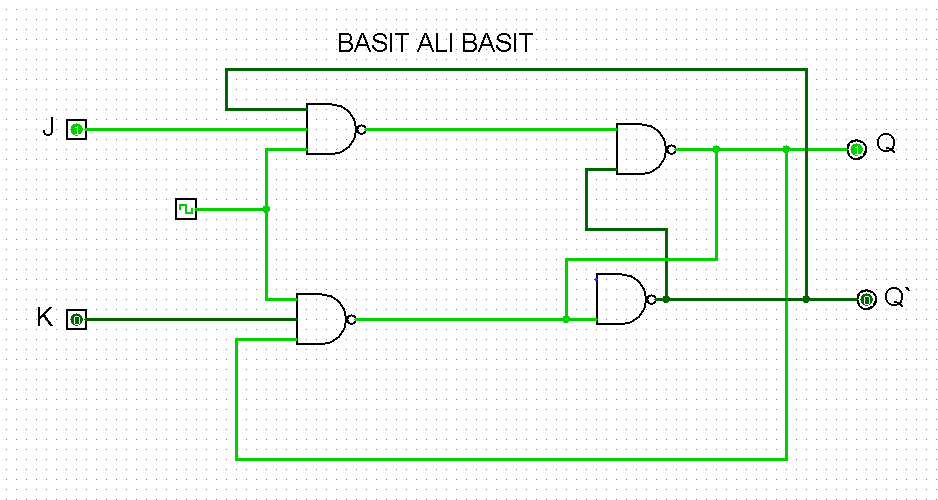
**J-K Flip Flops:**

**a. J-K Flip Flop Using NAND Gates:**

**Purpose:**

* A JK flip-flop is a type of digital flip-flop that is commonly implemented using NAND gates. The JK flip-flop has two inputs, J (set) and K (reset), and two outputs, Q and Q' (complement of Q). It is known for its versatility and ability to toggle its output based on certain conditions.
* The primary purpose of a JK flip-flop is to store one bit of information, similar to other flip-flops like the SR (Set-Reset) flip-flop. However, the JK flip-flop introduces a more flexible behaviour due to its ability to toggle its output under certain conditions.
* The NAND gate implementation of the JK flip-flop is common due to its simplicity. It typically involves using a pair of cross coupled NAND gates, similar to how an SR flip-flop is constructed. The JK flip-flop is widely used in digital circuits for various applications, including memory storage, sequential logic, and counter circuits. Its ability to toggle state makes it a valuable component in digital design.

**Diagram:**



**Working:**

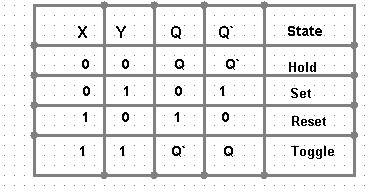
* When the J input is activated (high or 1) while the clock signal transitions from low to high (on the rising edge), it sets the Q output to 1. If the clock input is low or remains unchanged, the J input doesn't affect the flip-flop.
* When the K input is activated (high or 1) while the clock signal transitions from low to high, it resets the Q output to 0. Like the J input, if the clock input is low or remains unchanged, the K input doesn't affect the flip-flop.
* If both J and K inputs are activated (high or 1) while the clock signal transitions from low to high, the JK flip-flop toggles its state. If Q is 0, it becomes 1, and if Q is 1, it becomes 0.

**Procedure:**

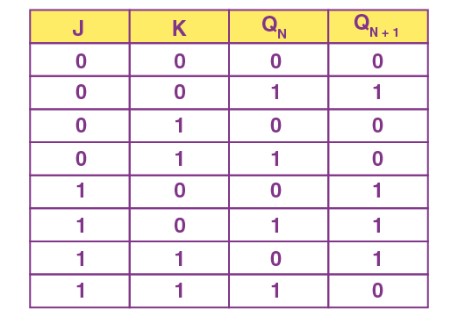
Creating a J-K flip flop using NAND gates involves following steps:

* Take a two NAND Gates which take three, three inputs name as J, K, Clock (CLK) and next output.
* Clock will always 1.
* Take two NAND gates more, the previous output will give as input and the next output will also give as inputs  Q and Q` will be the final Outputs.

**Truth Table:**



**Characteristics Table:**



**Experiment 12:**

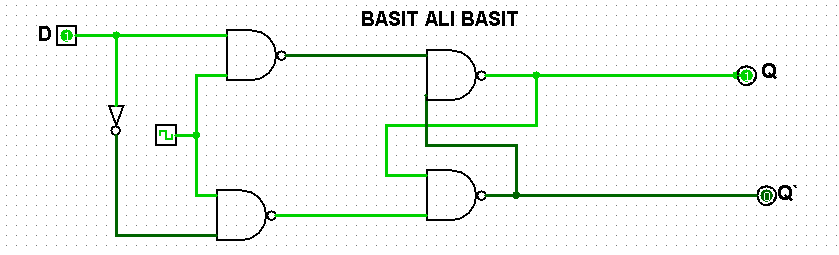
**D and T Flip Flop:**

**D Flip Flop Using NAND Gates:**

**Purpose:**

* + A D flip-flop is a type of flip-flop or bi stable multi vibrator that can store a single bit of data. It has two inputs: a data input (D) and a clock input (usually denoted as CLK or CP). The purpose of a D flip-flop implemented using NAND gates is to provide a simple and effective way to store and transfer binary information in digital circuits.
  + The D flip-flop is commonly used in digital circuits for various purposes, such as storage elements in registers, memory cells, and as building blocks for sequential logic circuits. It is particularly useful when a stable data state is required to be stored and updated based on a clock signal.
  + The NAND gate implementation is chosen for its simplicity and versatility in digital circuit design. It provides a straightforward way to create the required logical functions for a D flip-flop using a minimal number of components. The use of NAND gates is also practical in integrated circuit design where NAND gates are readily available.

**Diagram:**



**Working:**

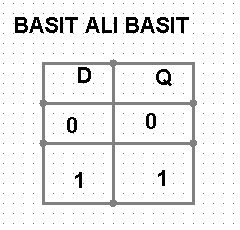
* + The D input is the data to be stored. When the clock signal transitions from low to high (on the rising edge or falling edge, depending on the specific implementation), the D flip-flop captures the data input and stores it.
  + The clock input determines when the D flip-flop captures the data. The data is transferred and stored only when the clock signal changes state (rising or falling edge, depending on the design).
  + In a D flip-flop using NAND gates, the basic configuration consists of two cross-coupled NAND gates. The output of one NAND gate is connected to one of the inputs of the other, creating a feedback loop. The data input and its complement are fed into the NAND gates, controlling the state of the flip-flop based on the clock signal.

**Procedure:**

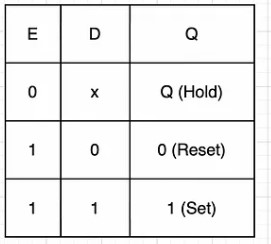
Creating a D flip flop using NAND and NOT gates involves following steps:

* + Take a two NAND Gates and one NOT Gate which take two inputs name as D, D` and Clock (CLK).
  + Clock will always 1.
  + Take two NAND gates more, the previous output will give as input and the next output will also give as inputs  Q and Q` will be the final Outputs.

**Truth Table:**



**Characteristics Table:**

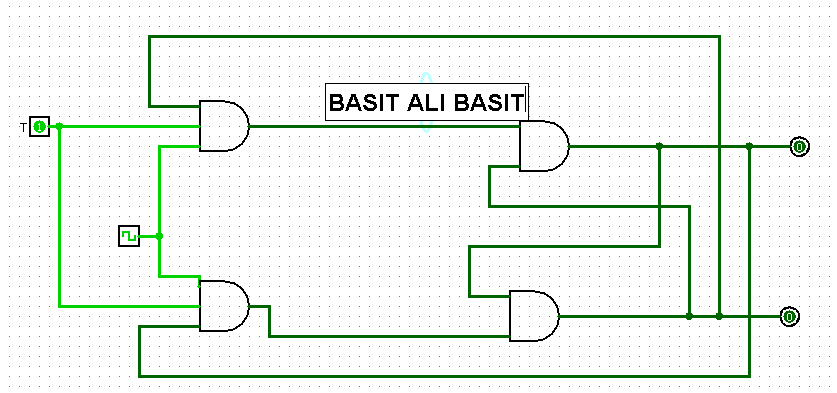


**T Flip Flop Using NAND and NOR Gates:**

**Purpose:**

* The purpose of T flip-flop is to toggling its output state based on the T (toggle) input and the clock signal. Combining both NAND and NOR gates might be done for specific design considerations, such as optimizing for certain performance characteristics or achieving a particular logic function.
* The primary purpose is to provide a flip-flop with a T input that toggles the output state based on the clock signal. Combining NAND and NOR gates might be done for specific design goals, such as optimizing for power consumption, propagation delay, or other factors.
* It's worth noting that the specific implementation details would depend on the exact logic design requirements and constraints. While NAND and NOR gates can be interchanged in many cases due to De Morgan's theorem, using both in the same circuit might have specific reasons related to the overall system design.

**Diagram:**



**Procedure:**

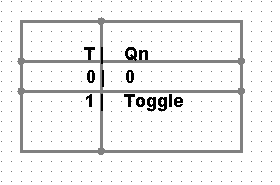
Creating a T flip flop using NAND and NOR gates involves following steps:

* Take a two NAND Gates which take two inputs name as T and Clock (CLK).
* Clock will always 1.
* Take two NOR gates, the previous output will give as input and the next output will also give as inputs  Q and Q` will be the final Outputs.

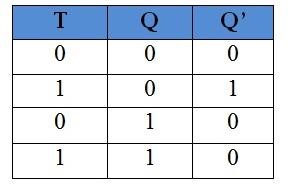
**Working:**

* The T input is the control input. When T is active (high or 1), the flip-flop toggles its state on the next clock pulse transition.
* The clock input determines when the T flip-flop captures the T input and toggles its state. The toggle operation typically occurs on a specific edge of the clock signal (rising or falling edge).
* The combination of NAND and NOR gates can be used to create the required logical functions for a T flip-flop. For instance, the T input and its complement might be fed into a NAND gate and a NOR gate, respectively. The outputs of these gates can then be combined to achieve the desired toggling behavior.

**Truth Table:**



**Characteristic Table:**



(------------------THE END-----------------)