Government of Gujarat

L. D. College of Engineering

LABORATORY MANUAL

Information Technology Department

Semester IV

3140707 -Computer Organization and Architecture

Year 2024-25

[Kale Sonali Parshuram - 240283116017]

L. D. College of Engineering
Ahmedabad -380015

Government of Gujarat

L. D. College of Engineering

CERTIFICATE

This is to certify that Ms.Kale Sonali Parshuram Enrollment No. 240283116017 of Semester IV Information Technology has successfully completed the prescribed term work/laboratory work of Computer Organization and Architecture (3140707) course within the four walls of L. D. College of Engineering. This is required as a partial fulfillment of the said course of Gujarat Technological University.

Date: _____

Course In-Charge *HOD*

Government of Gujarat

L. D. College of Engineering

Index

Exp. No.	Title	Date	Page No.	Initial of Course In-charge
1	Design the circuit of Half Adder and Half Subtractor.			
2	Design the circuit of Full Adder and Full Subtractor.			
3	Design the circuit of 4-bit Parallel Binary Adder.			
4	Design the circuit of 4-bit adder-subtractor.			
5	Design the circuit of 4-bit binary incrementer and decrementer.			
6	Design 4-bit Arithmetic circuit.			
7	Design one stage of Logic circuit.			
8	Design 4 bit combinational circuit shifter.			
9	Design 4 bit Arithmetic Logic Shift Unit.			

Government of Gujarat

L. D. College of Engineering

Index

Exp. No.	Title	Date	Page No.	Initial of Course In-charge
10	Develop Assembly language program to find out the 2's complement of the number.			
11	Develop Assembly language program to find out the subtraction of two numbers using indirect mode.			
12	Develop Assembly language program for the following pseudo code SUM=0 SUM=SUM+A+B DIF=DIF-C SUM=SUM+DIF			
13	Develop Assembly language program to add 100 numbers.			
14	Develop Assembly language program to logically OR two numbers.			
15	Develop Assembly language program to multiply two numbers by repeated addition method. (5*4=5+5+5+5)			
16	Develop Assembly language program to multiply two positive numbers.			

Experiment-1

AIM:- Design the circuit of Half Adder and Half Subtractor.

1) Half Adder

Theory:

- A **Half Adder** is a fundamental combinational logic circuit designed to add two single-bit binary numbers.
- It produces two outputs:
  - **Sum (S)**: Represents the least significant bit of the addition result.
  - **Carry (C)**: Represents the carry-out bit, which is passed to the next higher bit in case of multi-bit binary addition..
- The circuit operates purely combinationally, meaning it does not require any memory elements to store values.
- It is essential in digital electronics and is commonly used in the design of Arithmetic Logic Units (ALUs) and microprocessors.
- The Half Adder is an essential component in building more complex adders, such as the Full Adder, for multi-bit binary addition.

Boolean Expressions:

- **Sum (S)** = $A \oplus B$ (XOR operation)
- **Carry (C)** = $A \cdot B$ (AND operation)

Logic Circuit Components:

- **XOR Gate**: Generates the Sum output, providing a result of 1 when only one of the inputs is 1 (i.e., exclusive OR).
- **AND Gate**: Generates the Carry output, producing a 1 when both inputs are 1.

These two basic gates (AND and XOR) are sufficient to implement a Half Adder.

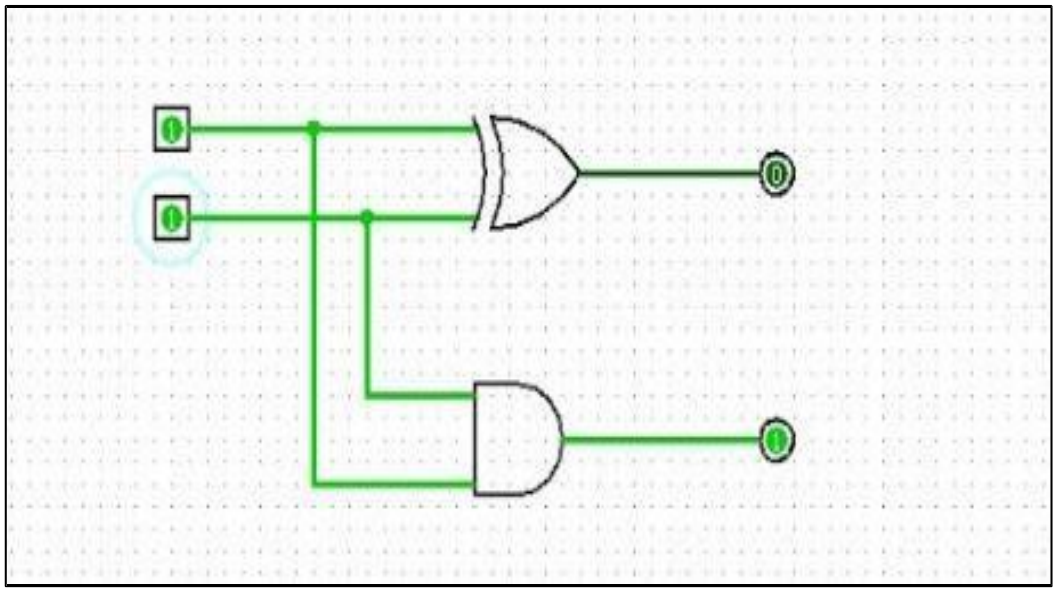
Applications of Half Adder:

- **Binary Addition**: The Half Adder is widely used in digital electronics to perform basic binary addition.
- **Building Block for Full Adder**: It serves as the foundation for more complex adder circuits, such as the Full Adder, which handles multi-bit binary numbers.
- **Microprocessor and ALU Design**: Half Adders are commonly used in processors and arithmetic circuits for performing basic addition tasks.
- **Counters and Shift Registers**: They help in designing circuits like counters, shift registers, and other arithmetic components.

Truth Table:

Input		Output	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Circuit/Program Implementation:-



2) Half Subtractor:

Theory:

- A **Half Subtractor** is a fundamental combinational logic circuit designed to subtract two single-bit binary numbers.
- It has two inputs:
  - **Minuend (A)**: The number from which subtraction is performed.
  - **Subtrahend (B)**: The number that is subtracted from the minuend.
- The circuit produces two outputs:
  - **Difference (D)**: Represents the result of the subtraction of BBB from AAA. It is the least significant bit of the result.
  - **Borrow (Bout)**: Indicates if a borrow is needed from the next higher bit during subtraction. A borrow occurs when AAA is smaller than BBB.
- The Half Subtractor is a basic circuit used for binary subtraction and is essential for building more complex subtractors, such as the Full Subtractor, which can handle multi-bit binary numbers.

Boolean Expressions:

- **Difference (D)** = A ⊕ B (XOR operation)
- **Borrow (Bout)** = A¯\overline{A}A · B (AND operation with NOT on AAA)

Logic Circuit Components:

- **XOR Gate**: Used to generate the **Difference** output. The XOR gate outputs 1 when exactly one of the inputs is 1, which represents the difference between the minuend and the subtrahend.
- **AND Gate and NOT Gate**: Used to generate the **Borrow** output. The AND gate, followed by a NOT gate on the minuend, produces a borrow when the minuend (AAA) is smaller than the subtrahend (BBB).

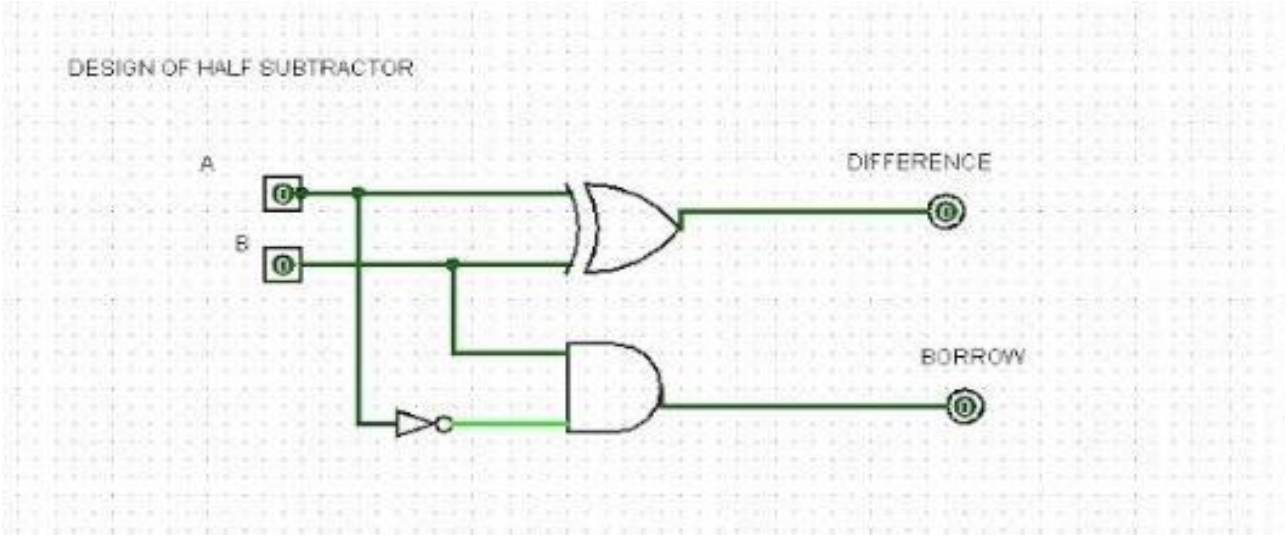
Applications of Half Subtractor:

- **Binary Subtraction**: The Half Subtractor is used in digital circuits to perform basic binary subtraction.
- **Building Block for Full Subtractor**: It serves as the foundation for more complex subtractor circuits, such as the Full Subtractor, which can handle multi-bit binary subtraction.
- **Arithmetic Logic Units (ALUs)**: Half Subtractors are often used in ALUs and microprocessors for basic subtraction operations.

Truth Table :

A	B	Difference (D)	Borrow (B)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Circuit/Program Implementation:-



Experiment-2

AIM:- Design the circuit of Full Adder and Full Subtractor.

1)Full Adder :

Theory:

- A **Full Adder** is a combinational logic circuit used to add three binary inputs:
  - Two bits AAA and BBB
  - A carry input CinCinCin from the previous stage.
- It produces two outputs:
  - **Sum (S)**: The least significant bit of the result.
  - **Carry-out (Cout)**: The carry to the next higher bit.
- Unlike the Half Adder, the Full Adder accounts for a carry input, making it ideal for multi-bit binary addition.

Boolean Expressions:

- **Sum (S)** = A⊕B⊕CinA \oplus B \oplus CinA⊕B⊕Cin (XOR operation)
- **Carry-out (Cout)** = (A·B)+(Cin·(A⊕B))(A \cdot B) + (Cin \cdot (A \oplus B))(A·B)+(Cin·(A⊕B)) (Combination of AND, XOR, and OR operations)

Logic Circuit Components:

- Implemented using **two Half Adders** and an **OR gate**.
- Requires:
  - **Two XOR Gates** for the sum.
  - **Two AND Gates** for intermediate carry values.
  - **One OR Gate** to produce the final carry-out.

Applications:

- Used in **multi-bit binary addition** in digital circuits.
- Forms the building block of **microprocessors** and **ALUs**.
- Used in **ripple carry adders** for multi-bit addition.
- Essential for **high-speed arithmetic operations** in computers.

Limitations:

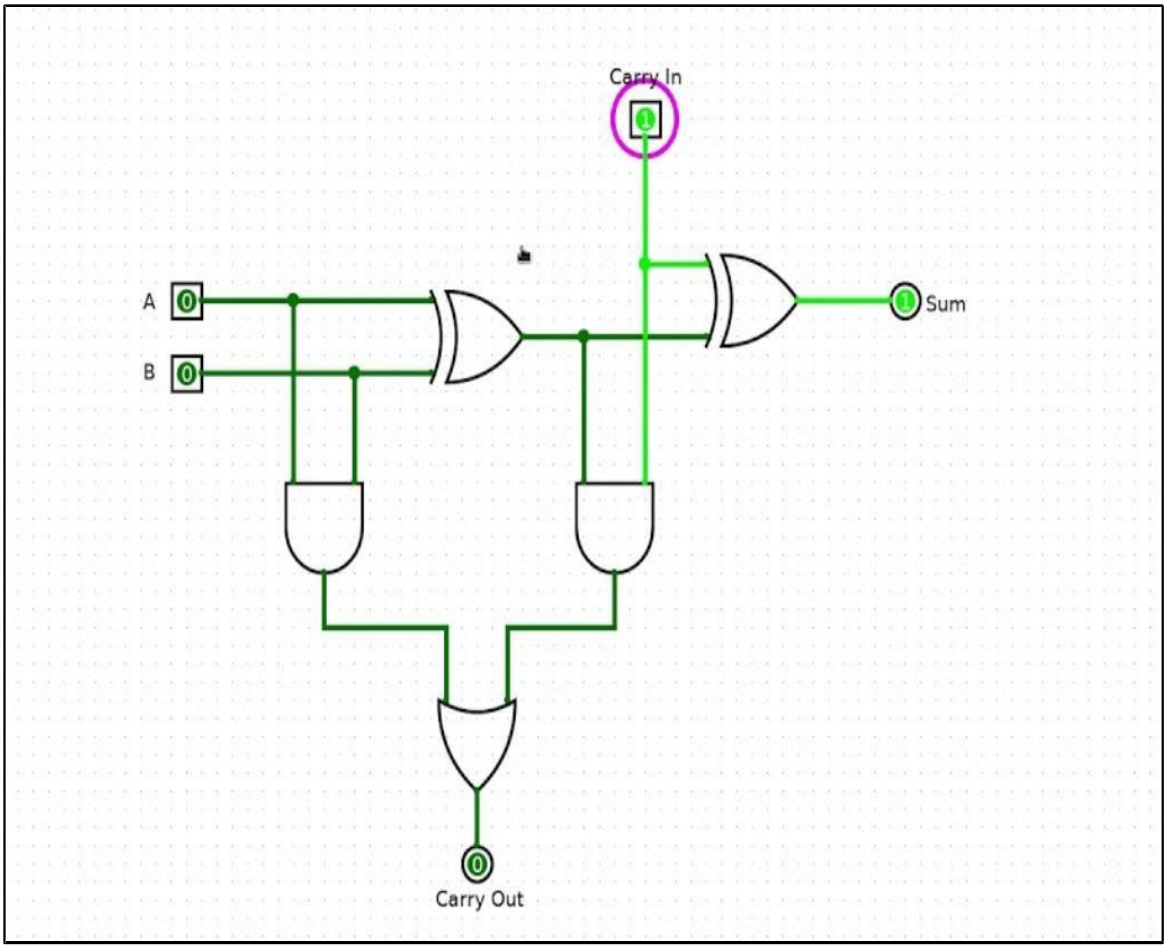
- **Ripple carry adders** can be slow when multiple Full Adders are cascaded.
- More complex than the Half Adder due to extra logic gates.
- Faster adders like **Carry Look-Ahead Adders** are used for better performance in large-scale operations.

Truth Table:-

Full Adder

A	B	C	C Out	Sum
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

Circuit Representation/Diagram:



2) Full Subtractor:

Theory:

- A **Full Subtractor** is a combinational logic circuit that performs the subtraction of three binary inputs:
  - **Minuend (A)**: The number from which subtraction is performed.
  - **Subtrahend (B)**: The number to be subtracted.
  - **Borrow-in (Bin)**: Borrow from the previous lower bit.
- It produces two outputs:
  - **Difference (D)**: The result of the subtraction.
  - **Borrow-out (Bout)**: Indicates if a borrow is needed for the next higher bit.

Boolean Expressions:

- **Difference (D)** = A⊕B⊕BinA \oplus B \oplus BinA⊕B⊕Bin (XOR operation)
- **Borrow-out (Bout)** = (A¯·B)+((A⊕B)¯·Bin)(\overline{A} \cdot B) + (\overline{(A \oplus B)} \cdot Bin)(A·B)+((A⊕B)·Bin) (AND, XOR, and OR operations)

Logic Circuit Components:

- **Two XOR Gates** to generate the Difference.
- **Two AND Gates** and **One OR Gate** to generate the Borrow-out.

Applications:

- Used in **multi-bit binary subtraction** in digital circuits.
- Essential in **microprocessors** and **ALUs**.
- Used in designing **counters**, **digital calculators**, and **control units**.
- Helps in **computational tasks** requiring subtraction.

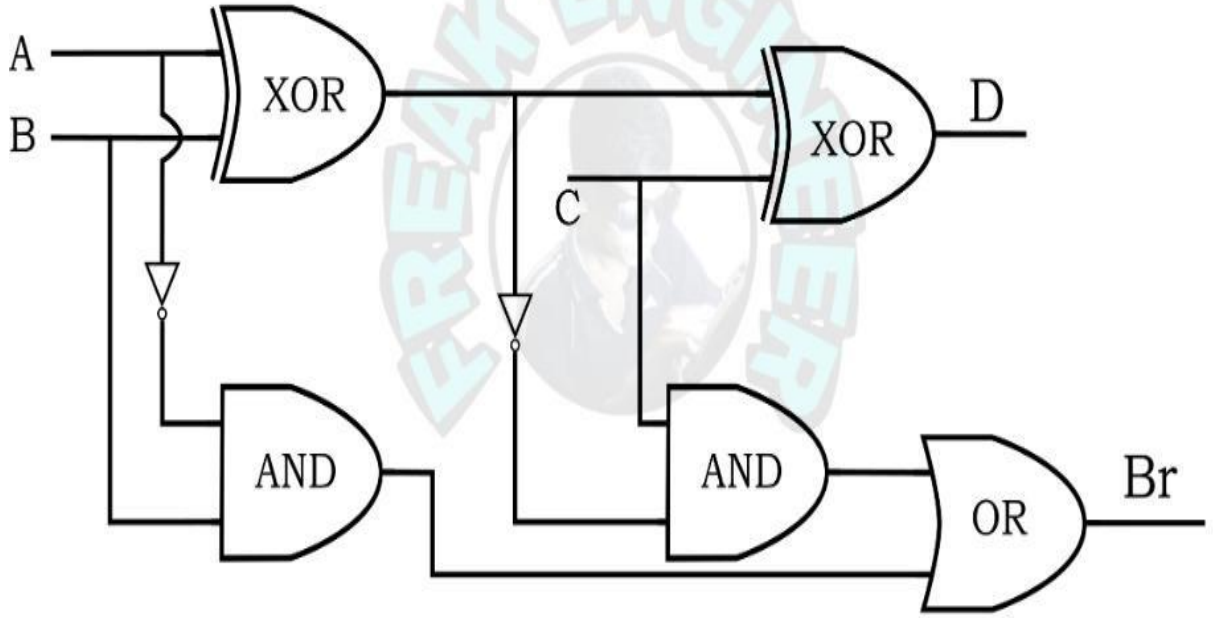
Limitations:

- More complex than a Half Subtractor due to the additional borrow input.
- **Cascading full subtractors** can slow down computation speed and increase **propagation delay**.
- Not suitable for high-speed applications requiring faster subtraction methods, such as **carry-lookahead subtractors**.

Truth Table:

A	B	Bin	Difference (D)	Borrow (Bout)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Circuit Representation/Diagram:



Experiment-3

AIM:- Design the circuit of 4-bit Parallel Binary Adder.

Theory:

- A **4-bit parallel binary adder** is a combinational circuit that adds two 4-bit binary numbers simultaneously.
- It consists of **four full adders** connected in cascade, where the carry output of one adder serves as the carry input to the next.
- The circuit produces a **4-bit sum** and a **carry-out (Cout)**, which can be used for further addition in multi-bit systems.

Working Principle:

- Addition begins from the **least significant bit (LSB)** and moves toward the **most significant bit (MSB)**.
- Each full adder adds corresponding bits and the carry input from the previous stage.
- The process continues until all bits are added and the final carry-out is generated.

Boolean Expressions:

- **Sum (Si)** = Ai⊕Bi⊕CinA_i \oplus B_i \oplus C_{in}Ai⊕Bi⊕Cin
- **Carry-out (Ci+1)** = (Ai·Bi)+(Cin·(Ai⊕Bi))(A_i \cdot B_i) + (C_{in} \cdot (A_i \oplus B_i))(Ai·Bi)+(Cin·(Ai ⊕Bi))

Circuit Components:

- **Four Full Adders**: Each adder adds two bits with the carry input from the previous stage.
- **Carry Propagation**: Carry output of one adder is connected to the carry input of the next adder.

Applications:

- Used in **microprocessors** and **ALUs** for arithmetic operations.
- Essential for **digital calculators** and **data processing systems**.
- Forms the basic unit for multi-bit binary adders in computational circuits.

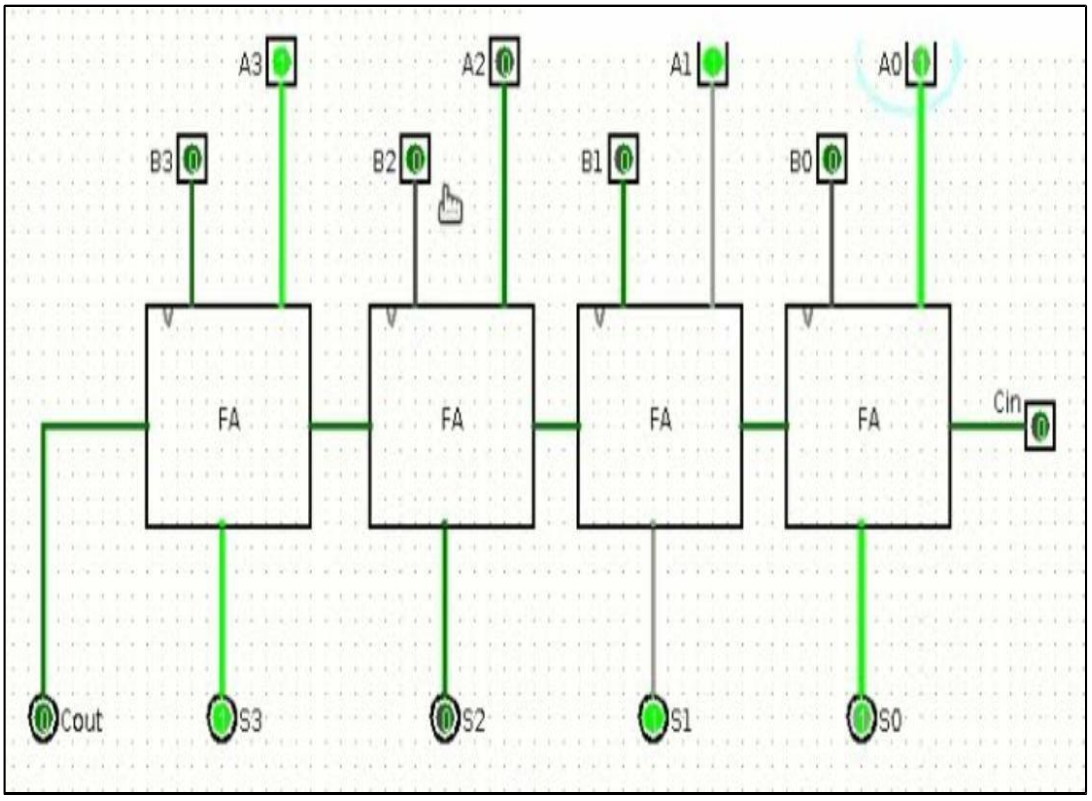
Limitations:

- **Propagation delay**: The carry propagation slows down computation.
- **Limited to 4-bit addition**: Cascading multiple adders is required for larger numbers.
- **Increased complexity**: More bits require additional adders, increasing circuit size.
- **Carry propagation delay**: Affects high-speed operations, addressed by **carry-lookahead adders**.

Truth Table:

C_{n-1}	A_n	B_n	Σ_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Circuit Representation/Diagram:



Experiment-4

AIM:- Design the circuit of 4-bit adder-subtractor.

Theory:

- A **4-bit adder-subtractor** is a combinational circuit that can perform both addition and subtraction of two 4-bit binary numbers.
- It uses a 4-bit parallel adder with an additional **mode control (M)** input to switch between addition and subtraction:
    - **M = 0**: Performs binary addition.
    - **M = 1**: Performs binary subtraction using the two's complement method (invert B and add 1).

Working Principle:

- The circuit consists of **four full adders**.
- **In addition mode**: Adds the binary numbers.
- **In subtraction mode**: The second operand (B) is complemented (inverted), and 1 is added (two's complement method).

Boolean Expressions:

- **Sum/Difference (Si)** = Ai⊕(Bi⊕M)⊕CinA_i \oplus (B_i \oplus M) \oplus C_{in}Ai⊕(Bi⊕M)⊕Cin
- **Carry/Borrow-out (Ci+1)** = (Ai·(Bi⊕M))+(Cin·(Ai⊕(Bi⊕M)))(A_i \cdot (B_i \oplus M)) + (C_{in} \cdot (A_i \oplus (B_i \oplus M)))(Ai·(Bi⊕M))+(Cin·(Ai⊕(Bi⊕M)))

Circuit Components:

- **Four full adders** for binary addition and subtraction.
- **XOR gates** to invert B during subtraction.
- **Mode control (M)** to toggle between addition and subtraction.

Applications:

- Used in **ALUs (Arithmetic Logic Units)** for both addition and subtraction.
- Essential in **microprocessors**, **digital calculators**, and **data processing systems**.
- Applied in **binary arithmetic** circuits and **computational applications**.

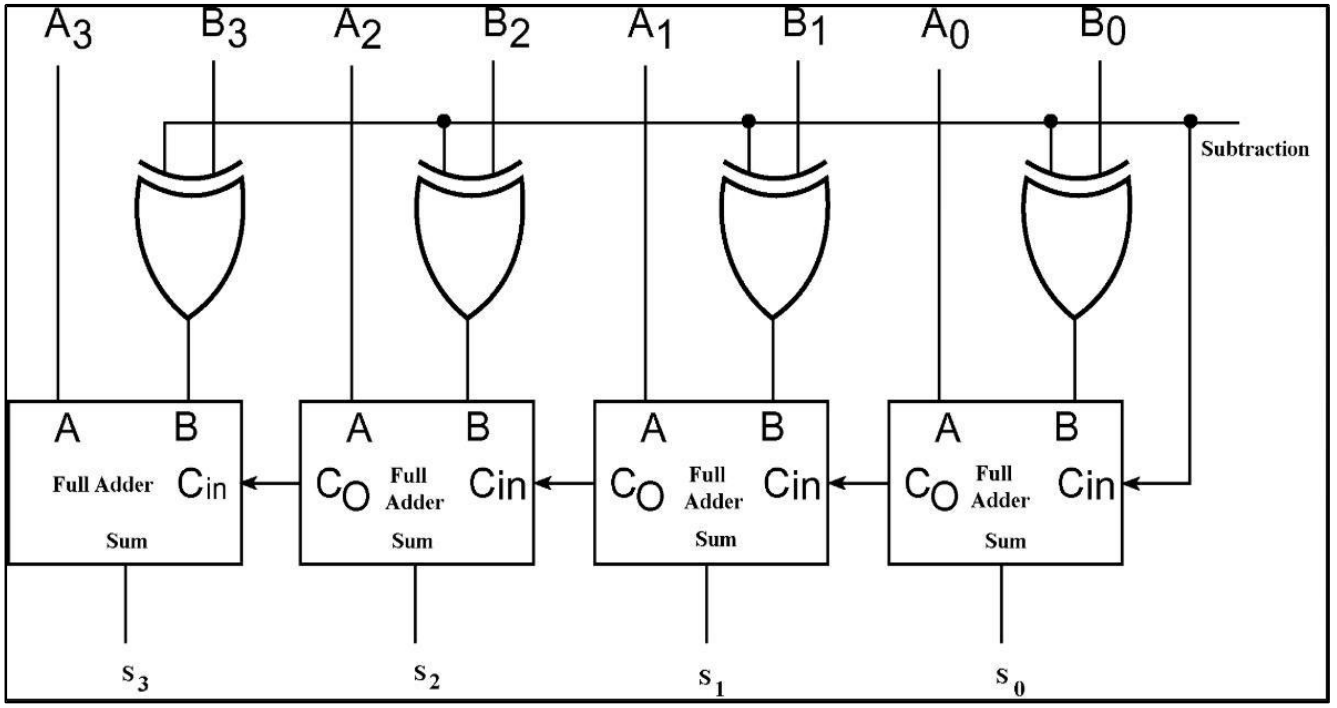
Limitations:

- **Propagation delay**: Carry or borrow must propagate through all full adders, slowing computation.
- **Limited to 4-bit operations**: Larger numbers need cascaded adders, increasing complexity.
- **Increased hardware complexity**: Requires XOR gates and mode control.
- **Overflow issues**: Overflow can occur if the result exceeds 4 bits.
- **Carry propagation delay**: Affects performance, mitigated by using techniques like **carry-lookahead adders**.

Truth Table:

	B3	B2	B1	B0	M	X3	X2	X1	X0
Addition	0	0	0	0	**0**	0	0	0	0
	0	0	0	1	**0**	0	0	0	1
	0	0	1	0	**0**	0	0	1	0
	0	0	1	1	**0**	0	0	1	1
	0	1	0	0	**0**	0	1	0	0
	0	1	0	1	**0**	0	1	0	1
	0	1	1	0	**0**	0	1	1	0
	0	1	1	1	**0**	0	1	1	1
	1	0	0	0	**0**	1	0	0	0
	1	0	0	1	**0**	1	0	0	1
Subs traction	0	0	0	0	**1**	1	0	0	1
	0	0	0	1	**1**	1	0	0	0
	0	0	1	0	**1**	0	1	1	1
	0	0	1	1	**1**	0	1	1	0
	0	1	0	0	**1**	0	1	0	1
	0	1	0	1	**1**	0	1	0	0
	0	1	1	0	**1**	0	0	1	1
	0	1	1	1	**1**	0	0	1	0
	1	0	0	0	**1**	0	0	0	1
	1	0	0	1	**1**	0	0	0	0

Circuit Representation/Diagram:



Experiment-5

AIM: Design the circuit of 4-bit binary incrementer and decrementer.

4-Bit Binary Incrementer

Theory:

- A **4-bit binary incrementer** is a combinational circuit used to increase a 4-bit binary number by 1.
- It is widely used in digital systems, such as in **counters**, **address generation**, and **arithmetic operations**.

Working Principle:

- The circuit adds 1 to a 4-bit binary input.
- Instead of using a full adder, a simplified logic design is used to reduce complexity.
- The **least significant bit (LSB)** toggles (flips), and carry is propagated through higher bits until a 0 is encountered, mimicking binary addition but optimized for a simple increment by 1.

Boolean Expressions:

- **For the least significant bit (S0)**:
  - S0=A0⊕1S_0 = A_0 \oplus 1S0=A0⊕1
  - **Carry (C1)**: C1=A0·1C_1 = A_0 \cdot 1C1=A0·1
- **For higher bits**:
  - Si=Ai⊕CiS_i = A_i \oplus C_iSi=Ai⊕Ci
  - **Carry (Ci+1)**: Ci+1=Ai·CiC_{i+1} = A_i \cdot C_iCi+1=Ai·Ci

Circuit Components:

- **XOR gates**: Used to compute the sum for each bit.
- **AND gates**: Used to generate the carry for higher bits.

Applications:

- **Counters** to increment values.
- **Address generation** for memory access.
- **ALUs (Arithmetic Logic Units)** in processors.
- **Control logic** for sequential circuits.

Limitations:

- **Limited to 4-bit increments**: Larger numbers need multiple incrementers.
- **Propagation delay** increases with more bits.
- **Cannot handle decrement operations**, requiring a separate circuit for subtraction.
- **Overflow** occurs when the input is 111111111111, causing it to roll over to 000000000000.

Truth Table:

Current state				Next state			
D3	D2	D1	D0	D3(next)	D2(next)	D1(next)	D0(next)
0	0	0	0	0	0	1	0
0	0	0	1	0	0	1	1
0	0	1	0	0	1	0	0
0	0	1	1	0	1	0	1
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	1	0	0	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	1	0
1	0	0	1	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	1	1	0
1	1	0	1	1	1	1	1
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	1

Circuit Representation/Diagram

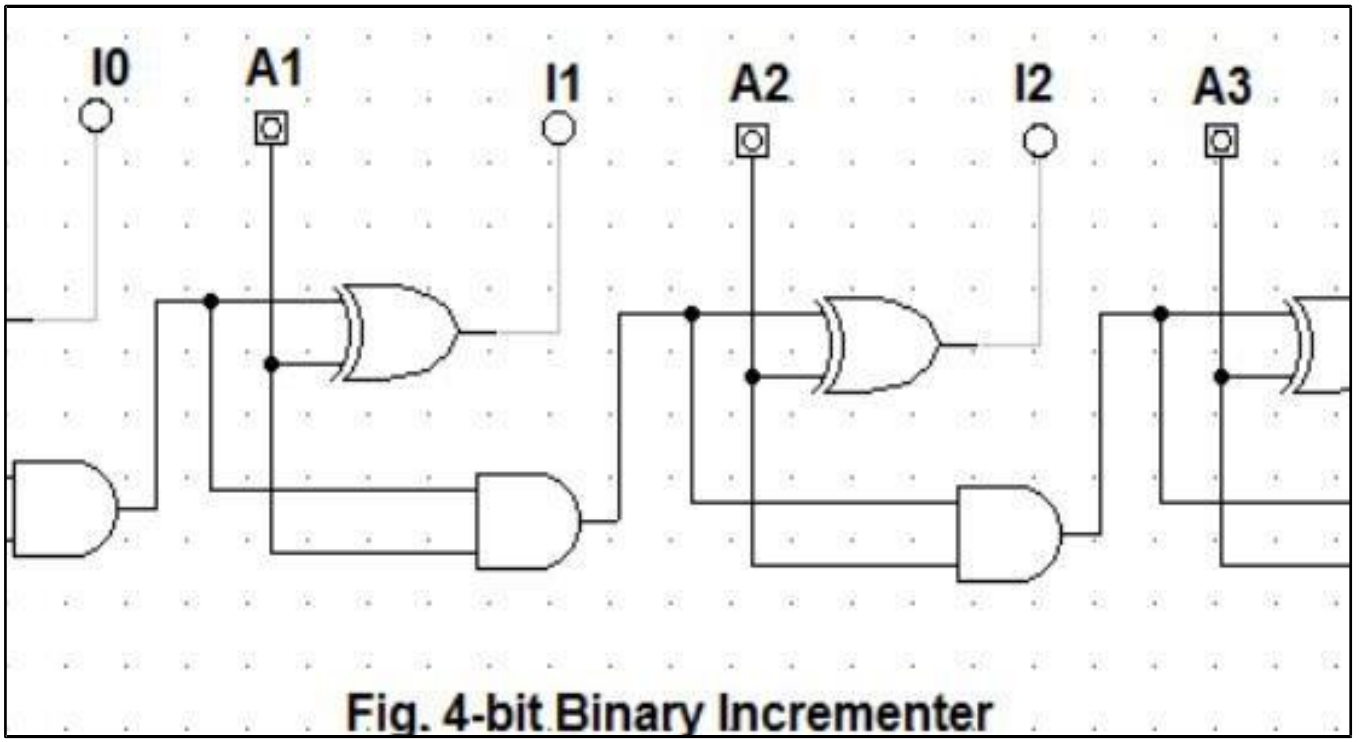


Fig. 4-bit Binary Incrementer

4-Bit Binary Decrementer

Theory:

- A **4-bit binary decrementer** is a combinational circuit used to decrease a 4-bit binary number by 1.
- It is commonly used in digital systems for **countdown operations**, **memory address decrementing**, and in **arithmetic logic units (ALUs)**.

Working Principle:

- The circuit takes a 4-bit binary input and subtracts 1.
- Instead of using a full subtractor, a simplified logic design is used to reduce complexity.
- The **least significant bit (LSB)** toggles, and a borrow is propagated through higher bits until a 1 is encountered.
- This process mimics binary subtraction but is optimized for decrementing by 1.

Boolean Expressions:

- **For the least significant bit (D0)**:
  - D0=A0⊕1D_0 = A_0 \oplus 1D0=A0⊕1
  - **Borrow (B1)**: B1=A0¯·1B_1 = \overline{A_0} \cdot 1B1=A0·1
- **For higher bits**:
  - Di=Ai⊕BiD_i = A_i \oplus B_iDi=Ai⊕Bi
  - **Borrow (B_{i+1})**: Bi+1=Ai¯·BiB_{i+1} = \overline{A_i} \cdot B_iBi+1=Ai·Bi

Circuit Components:

- **XOR gates**: Used to compute the difference for each bit.
- **AND gates**: Used to generate the borrow for higher bits.

Applications of 4-bit decrementer

- Used in counters for countdown operations.
- Applied in address generation for stack memory.
- Forms a part of alus (arithmetic logic units) in processors.
- Helps in control logic for sequential circuits.

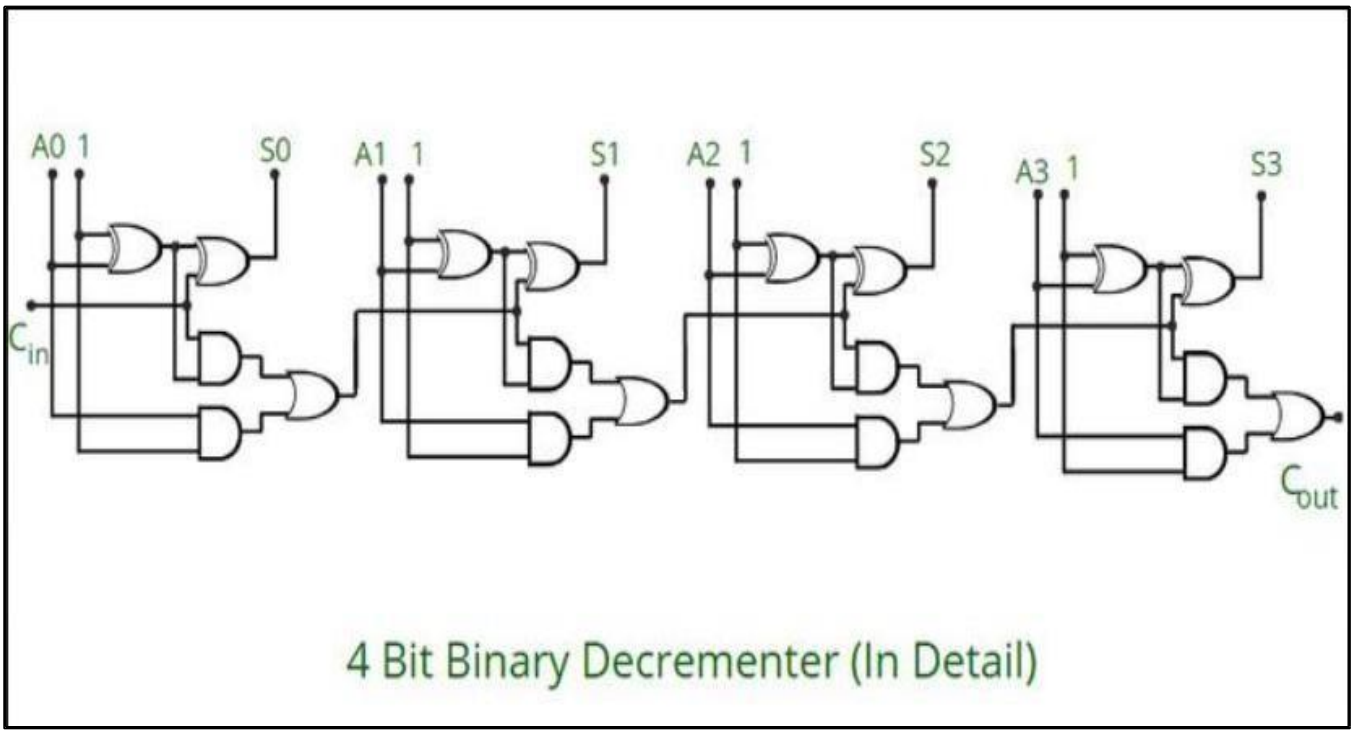
Limitations of 4-bit decrementer

- Limited to 4-bit decrements; for larger numbers, multiple decrementers are needed.
- Propagation delay increases as the number of bits increases.
- Cannot handle increment operations, requiring a separate circuit for addition.
- Underflow occurs when the input is at its minimum value (0000), rolling over to 1111.

Truth Table:

Input (A3 A2 A1 A0)	Output (A3 A2 A1 A0)
0000	1111 (Underflow)
0001	0000
0010	0001
0011	0010
0100	0011
0101	0100
0110	0101
0111	0110
1000	0111
1001	1000
1010	1001
1011	1010
1100	1011
1101	1100
1110	1101
1111	1110

Circuit Representation/Diagram:



Experiment-6

AIM: Design the circuit of 4-bit Arithmetic Circuit.
4-Bit Arithmetic Circuit

Theory: A **4-bit arithmetic circuit** is designed to perform basic arithmetic operations on 4-bit binary numbers, such as addition, subtraction, and logical functions.

Components:

1. **Full Adders (FAs)**: Used for binary addition.
2. **Multiplexers (MUXs)**: Used for selecting different operations.
3. **Logic Gates** (AND, OR, XOR, NOT): Control the circuit functionality.
4. **Control Lines**: Select the specific operation to be performed.

Operations:

1. **Addition (A + B)**: Adds two 4-bit binary numbers.
2. **Subtraction (A - B)**: Uses two's complement for binary subtraction.
3. **Increment (A + 1)**: Adds 1 to the 4-bit number.
4. **Decrement (A - 1)**: Subtracts 1 from the 4-bit number.
5. **Bitwise AND (A & B)**: Performs logical AND.
6. **Bitwise OR (A | B)**: Performs logical OR.
7. **Bitwise XOR (A ⊕ B)**: Performs logical XOR.
8. **Complement (A')**: Produces the one's complement.

Circuit Design:

- **Four Full Adders**: Each bit is added separately using full adders.
- **Multiplexer**: Selects between different operations based on control inputs.
- **Control Inputs**: Define the operation (addition, subtraction, etc.).

Working Principle:

- **Addition**: Directly adds A and B.
- **Subtraction**: Converts B to its two's complement and adds to A.
- **Increment/Decrement**: Adds or subtracts 1 to/from the number.
- **Bitwise Operations**: Performed using AND, OR, XOR gates.

Limitations:

1. **Limited Range**: Can only handle values between 0000 (0) and 1111 (15) in unsigned operations.
2. **Carry Propagation Delay**: Adds delay as carry propagates through full adders.
3. **Overflow**: Results exceeding 4-bits cause overflow.
4. **Fixed Bit Width**: Can't handle larger numbers without increasing bit width.
5. **Complexity**: More operations require additional components.
6. **No Multiplication/Division**: These operations require additional logic.

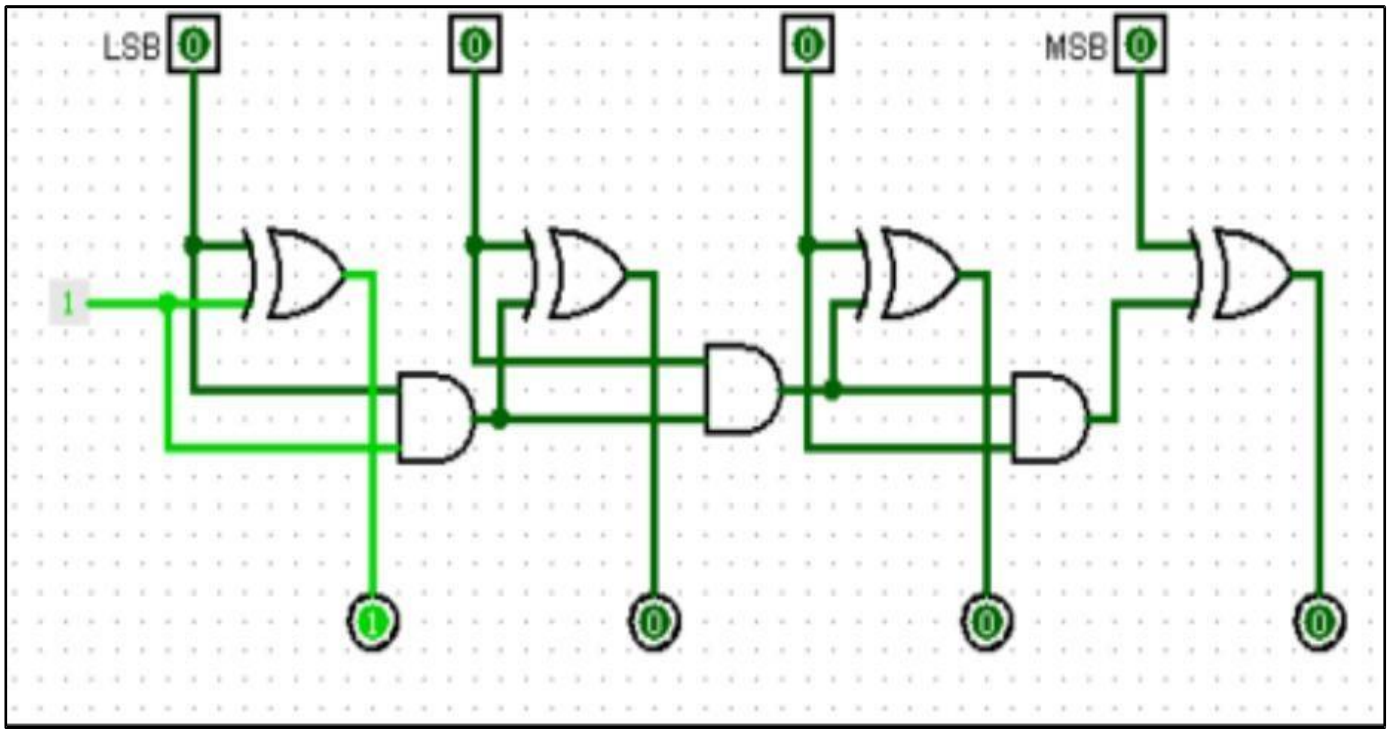
Conclusion:

- The **4-bit arithmetic circuit** is vital in digital systems, especially in **ALUs** within microprocessors, enabling basic arithmetic operations. However, it has limitations regarding range, speed, and the types operations it can handle, making it suitable mainly for simple tasks.

Truth Table:

S3	S2	S1	S0	Cin	Operation
0	0	0	0	0	A
0	0	0	0	1	A+1
0	0	0	1	0	A+B
0	0	0	1	1	A+B+1
0	0	1	0	0	A+B'
0	0	1	0	1	A+B'+1
0	0	1	1	0	A-1
0	0	1	1	1	A
0	1	0	0	x	A ∧ B
0	1	0	1	X	A v B
0	1	1	0	X	A⊕ B
0	1	1	1	X	A'
1	0	X	X	X	SHR
1	1	x	X	X	SHL

Circuit Representation / Diagram:



Experiment-7

AIM: Design one stage of Logic circuit.

Theory:

- A **one-state logic circuit** is a digital circuit designed to maintain a fixed state, either logic 0 (LOW) or logic 1 (HIGH). Unlike combinational or sequential circuits, it operates in a stable state, often used for functions like pull-up/pull-down, clock stabilization, and reset.

Components:

1. **Logic Gates** (AND, OR, NOT): Enforce the desired state.
2. **Resistors** (Pull-up/Pull-down): Maintain a stable logic level when no input is provided.
3. **Flip-flops/Latches**: Store and hold a fixed state in some circuits.
4. **Transistors**: Used as switches to maintain the defined state.

Working Principle:

- **Pull-up/Pull-down Resistors**:
  - Pull-up keeps the line at logic HIGH (1) when no input is present.
  - Pull-down keeps the line at logic LOW (0) when no input is present.
- **Latch Mechanisms**:
  - Simple latches (e.g., SR latch, D flip-flop) store a state and remain in it until reset.
- **Feedback Loops**:
  - Some circuits use feedback to maintain a constant output.

Applications:

1. **Reset Circuits**: Ensure systems start in a known state.
2. **Pull-up/Pull-down Networks**: Maintain default logic levels in open circuits.
3. **Clock and Timing Circuits**: Provide stable reference signals.
4. **Memory Hold Circuits**: Keep a fixed state until externally triggered.
5. **Power-on Initialization**: Ensures stable logic states during startup.

Limitations:

1. **Lack of Processing**: Cannot perform logic operations or complex tasks.
2. **Fixed Output**: Only holds one state, unsuitable for more dynamic functions.
3. **External Influence**: Noise or voltage fluctuations can disrupt the state.

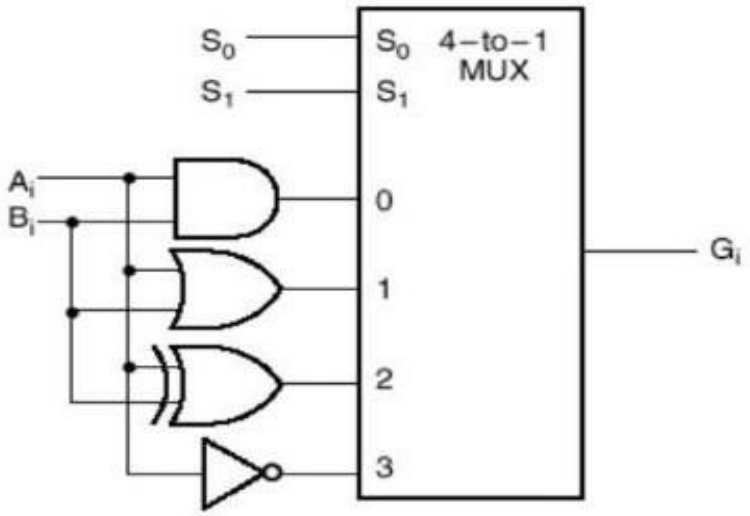
Conclusion:

- One-state logic circuits are essential for ensuring stability in digital systems. They don’t perform calculations but provide crucial support by maintaining defined states, preventing floating signals, and ensuring smooth system operation

Truth Table:

A	B	Cin	Sum (S)	Carry-out (Cout)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Circuit Representation / Diagram:



(a) Logic Diagram

Experiment-8

AIM: Design 4 bit combinational circuit shifter.
4-bit Combinational Circuit Shifter

Theory:

- A **4-bit combinational circuit shifter** is a digital circuit that shifts a 4-bit binary input left or right. It is a combinational circuit, meaning the output depends solely on the current input, with no memory elements involved. This type of circuit is commonly used in arithmetic operations, data manipulation, and logical processing.

Types of Shifting Operations:

1. **Logical Left Shift (LLS)**: Shifts all bits left, inserting 0 at the least significant bit (LSB).
2. **Logical Right Shift (LRS)**: Shifts all bits right, inserting 0 at the most significant bit (MSB).
3. **Arithmetic Right Shift (ARS)**: Shifts bits right, keeping the sign bit (MSB unchanged).
4. **Circular Left Shift (CLS)**: Shifts bits left, moving the MSB to the LSB.
5. **Circular Right Shift (CRS)**: Shifts bits right, moving the LSB to the MSB.

Circuit Components:

1. **Multiplexers (MUXs)**: Select between different shift operations.
2. **Logic Gates (AND, OR, NOT)**: Control the shifting process.
3. **Shift Register (in some designs)**: Manages bit positions during shifting.

Working Principle:

- The circuit takes a 4-bit input (A3, A2, A1, A0).
- A control signal (S) determines the type of shift operation.
- Depending on the control signal, logic gates and multiplexers reposition the bits.
- The result is displayed as a new 4-bit value after shifting.

Applications:

1. **Arithmetic Operations**: Used for multiplication/division by powers of 2.
2. **Data Manipulation**: Rearranges bits in digital processing.
3. **Cryptography/Encoding**: Applied in encryption and data scrambling.
4. **Microprocessors/ALUs**: Essential for arithmetic and logic unit operations.

Limitations:

1. **Loss of Bits**: Logical shifts may discard bits, leading to information loss.
2. **Limited Range**: Works with small 4-bit data sizes.
3. **No Storage**: As a combinational circuit, it doesn't retain previous inputs.

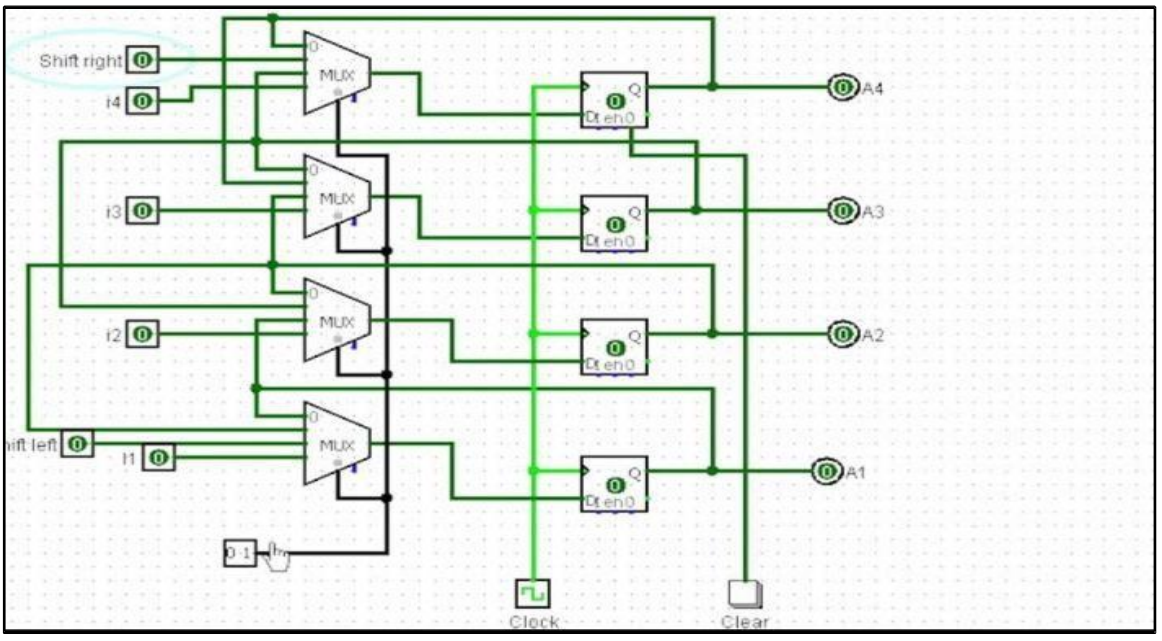
Conclusion:

- The 4-bit combinational circuit shifter is crucial in digital systems for performing shifting operations efficiently. While it’s ideal for basic shift operations, it has limitations in terms of data loss and the ability to handle larger data sizes.

Truth Table:

Iteration (clock tick)	In	Q1	Q2	Q3	Q4
0	0	0	0	0	0
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1
6	0	0	0	0	0
7	1	0	0	0	0
8	1	1	0	0	0
9	1	1	1	0	0
10	1	1	1	1	0
11	1	1	1	1	1

Circuit Representation/Diagram:



Experiment-9

AIM: Design 4 bit Arithmetic Logic Shift Unit.

4-bit Arithmetic Logic Shift Unit (ALSU)

Theory:

- A **4-bit Arithmetic Logic Shift Unit (ALSU)** is a combinational circuit that performs arithmetic, logical, and shift operations on a 4-bit input. It combines an **Arithmetic Logic Unit (ALU)** and a **Shifter** into a single unit, controlled by a selection input.

Components:

1. **Arithmetic Logic Unit (ALU):** Handles arithmetic (addition, subtraction) and logic operations (AND, OR, XOR, NOT).
2. **Shifter Unit:** Performs logical, arithmetic, and circular shifts.
3. **Multiplexer (MUX):** Selects between arithmetic, logic, and shift operations.
4. **Control Lines:** Used to choose the desired operation.

Operations:

- **Arithmetic:**
  - Addition (A + B)
  - Subtraction (A - B) using Two's Complement
- **Logical:**
  - Bitwise AND (A & B)
  - Bitwise OR (A | B)
  - Bitwise XOR (A $\oplus$ B)
  - Bitwise NOT (A')
- **Shift Operations:**
  - Logical Left Shift (LLS)
  - Logical Right Shift (LRS)
  - Arithmetic Right Shift (ARS)
  - Circular Left Shift (CLS)
  - Circular Right Shift (CRS)

Circuit Design:

- Full Adders for arithmetic.
- Logic gates (AND, OR, XOR, NOT) for logic operations.
- Shift registers or multiplexers for shifting.
- Multiplexer-based control system for operation selection.

Working Principle:

- Inputs A and B (4-bit) are provided to the ALSU.
- A selection input (S) determines the operation: arithmetic, logic, or shift.
- The chosen operation is executed, and the 4-bit result is output.

Applications:

- **Microprocessors & ALUs:** Used for processing arithmetic and logical instructions.
- **Digital Signal Processing (DSP):** Data manipulation and computation.
- **Encryption & Cryptography:** Bitwise transformations for secure data processing.
- **Memory and Registers:** Shifting and logical operations for data handling.

Limitations:

1. **Limited Bit-Width:** Works only with 4-bit data.
2. **No Storage Capability:** Does not retain past results (combinational circuit).
3. **Carry Propagation Delay:** Delay due to full adders in arithmetic operations.

Conclusion:

- The **4-bit ALSU** combines arithmetic, logical, and shift operations, enhancing the processing power of digital systems. It forms the foundation for computations in microprocessors and other digital devices.

Truth Table:

S_3	S_2	S_1	S_0	C_{in}
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
0	1	0	0	x
0	1	0	1	x
0	1	1	0	x
0	1	1	1	x
1	0	x	x	x
1	1	x	x	x

Circuit Representation / Diagram: