REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE SUPERIEUREEN SCIENCES APPLIQUEES
--TLEMCEN--

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
-تلمسان-

Higher School of Applied Sciences of Tlemcen

Second-Cycle Department

Practical Work Handout

# Digital Electronics

Prepared by:

Dr Mohammed M'HAMEDI

Dr Nadjia MEDJAHDI

# PREFACE

We are pleased to introduce this handbook, which brings together the practical workshop and exercises from the **"**Digital Electronics **"** module, specifically designed for first-year engineering students in the Automatic Control specialization**.**

This laboratory manual aims to help students acquire a solid understanding of digital logic fundamentals and their practical implementation using the LAB KL-300 system. Through a series of structured experiments, students will explore both combinational and sequential logic circuits, moving gradually from basic Boolean operations to more advanced systems such as arithmetic and counter circuits. Each practical session includes clear learning objectives**,** theoretical reminders**,** step-by-step experimental procedures, and analysis questions to reinforce understanding. The hands-on activities are intended to bridge the gap between theory and practice, encouraging analytical thinking and systematic problem-solving.

By completing these practical exercises, students will gain a solid foundation in digital logic analysis, circuit implementation, and systematic reasoning—key competencies for future courses in electronics, microcontrollers, and computer architecture. We hope that this handout will serve as a valuable companion throughout your laboratory sessions and foster both technical competence and curiosity in digital system design.

# Table of Contents

**Workshop 3: Combinational Circuits – Encoders, Decoders, and Multiplexers**



**Workshop 4: Sequential Circuits – Flip-Flops (RS, D, JK)**

**Workshop 5: Sequential Logic Circuits – Binary Counters and Down-counters**

# List of Figure

# INTRODUCTION

The **"**digital electronics**"** module is a fundamental component of the Automatic Control Engineering curriculum.

It introduces students to the basic principles that underlie all modern digital systems, from simple logic circuits to complex microprocessors. Understanding digital logic is essential for engineers who will later work with automation systems, embedded controllers, and programmable devices. This practical work handout is designed to accompany the theoretical course by offering direct, hands-on experience using the LAB KL-300 Digital Logic Training System.

The KL-300 laboratory bench provides a complete platform for studying digital electronics through its thirteen functional modules (KL33001 – KL33013), covering logic gates, combinational and sequential circuits, arithmetic units, and counters. Each practical session guides students through the construction, testing, and analysis of real hardware circuits using switches, LEDs, and logic indicators.

The practical program is organized into five main workshops preceded by an introductory session:

| **Workshop** | **Title** | **Main Objectives** |
|---|---|---|
| **Part 0** | Introduction and System Overview | Review Boolean algebra, familiarize students with the LAB KL-300 setup and basic wiring techniques. |
| **Workshop 1** | Basic Logic Gates | Study fundamental gates (AND, OR, NOT, NAND, NOR, XOR) and verify Boolean relationships experimentally. |
| **Workshop 2** | Combinational Circuits – Arithmetic Units | Design and test half and full adders, 4-bit parallel adders and subtractors, and simple arithmetic/logic circuits. |
| **Workshop 3** | Combinational Circuits – Encoders, Decoders, and Multiplexers | Explore information selection and coding circuits and analyze signal routing using multiplexers/demultiplexers. |
| **Workshop 4** | Sequential Circuits – Flip-Flops (RS, D, JK) | Observe memory behavior in digital circuits and understand synchronization with clock signals. |
| **Workshop 5** | Counters (Synchronous and Asynchronous) | Construct and analyze binary and decimal counters; study timing and propagation effects. |

Throughout these sessions, students are encouraged to observe carefully**,** record measurements**,** and interpret results in light of theoretical principles.

The goal is to develop not only manual skills and familiarity with laboratory instruments but also the ability to reason logically and systematically about digital circuit operation.

By the end of this laboratory module, students should be able to:

- Understand and apply Boolean logic in real hardware systems.
- Design, assemble, and test basic combinational and sequential circuits.
- Identify and correct logical or wiring errors during experimentation.
- Establish the foundation required for advanced courses in microcontrollers**,** automation**,** and embedded system design**.**

# Introduction and System Overview

## I. Introduction

Boolean algebra and logic gates form the foundation of digital electronics and computer systems. All modern digital devices, from simple calculators to advanced processors, rely on principles of Boolean logic to perform operations, make decisions, and process information.

Boolean algebra, introduced by George Boole in the mid-19th century, provides a mathematical framework for analyzing and simplifying logical expressions. Unlike conventional algebra, which deals with real numbers, Boolean algebra operates on binary variables that take only two values: **0 (false)** and **1 (true)**.

Logic gates are the physical implementation of Boolean functions in electronic circuits. They are the building blocks of digital hardware, used to design circuits that perform tasks such as addition, comparison, memory storage, and control operations. Common logic gates include **AND, OR, NOT, NAND, NOR, XOR, and XNOR**, each corresponding to a fundamental Boolean operation.

This chapter examines the fundamental rules of Boolean algebra, the properties that enable the simplification of expressions, and how these principles are implemented in logic gate circuits. Mastering these concepts is essential for understanding the design of digital systems, microprocessors, and computer architecture.

In this chapter, we will explore:

- **The Basics of Boolean Algebra:** The fundamental operators (AND, OR, NOT), Boolean expressions, and truth tables.
- **Logic Gates:** The symbols and behavior of the basic gates (AND, OR, NOT, NAND, NOR, XOR, XNOR) that perform Boolean operations.
- **Boolean Laws and Theorems:** The set of rules (such as Commutative, Associative, Distributive, De Morgan's Theorems) that allow us to manipulate and simplify complex logical expressions, leading to more efficient circuit designs.
- **Canonical Forms:** Standard ways of representing Boolean functions, such as the Sum-of-Products (SOP) and Product-of-Sums (POS) forms, which provide a systematic method for designing logic circuits.
- **Circuit Synthesis and Analysis:** The process of translating a real-world problem into a Boolean expression, designing its logic gate implementation, and analyzing its function.

## II. Boolean algebra and logic functions

To understand the fundamental principles of Boolean algebra, its operations, and how it is used to define, analyze, and simplify logic functions that form the basis of digital circuit design.

Boolean algebra is a branch of mathematics that deals with variables called Logical variable that can have only two possible values: TRUE (1) or FALSE (0). This binary nature makes it the perfect mathematical tool for analyzing and designing digital circuits, where signals are either a high voltage (~1) or a low voltage (~0).

Boolean algebra provides the rules, and logic gates provide the tools. Together, they allow us to:

1. **Formally describe the behavior of a digital system.**
2. **Analyze complex circuits by deriving their Boolean functions.**
3. **Simplify circuits to use the fewest possible gates, optimizing for cost, power, and speed.**
4. **Synthesize new circuits from a set of requirements.**

Algebraic representation is based on three things. The first is a logical variable, the second is a logical operator, and the third is a logical function.

### II.1. logical variable

Boolean algebra is a mathematical system that represents **logical operations** and relationships between binary variables. Each variable can take only two possible values:

- **1** → represents **True / High / ON**
- **0** → represents **False / Low / OFF**

Boolean expressions describe how variables combine using logical operators. These expressions can be represented in different ways:

### II.2. Logic operators

Logic operators, or Boolean operators, are symbols or words used in logic to combine or manipulate logic functions. There are three fundamental operations in Boolean algebra from which all others are built. Each operation has a corresponding **logic gate**.

- **The NOT** operator **(Inversion / Complement)**

  Symbol: $\overline{A}$

  Logic Gate: NOT gate

  Definition**:** The output is 1 **only if** the input is 0, and vice versa. It inverts the input.

**Truth Table:**

| A | NOT A ($\overline{A}$) |
|---|---|
| 0 | **1** |
| 1 | **0** |

- **The AND operator (Logical Multiplication)**

Symbol: A.B

Logic Gate: AND gate

Definition**:** The output is 1 only if all inputs are 1. If any input is 0, the output is 0.

**Truth Table:**

| A | B | A · B |
|---|---|---|
| 0 | 0 | **0** |
| 0 | 1 | **0** |
| 1 | 0 | **0** |
| 1 | 1 | **1** |

- **The OR operator (Logical Addition)**

Symbol: A+B

Logic Gate: OR gate

Definition: The output is 1 only if all inputs are 1. If any input is 0, the output is 0.

**Truth Table:**

| A | B | A + B |
|---|---|---|
| 0 | 0 | **0** |
| 0 | 1 | **1** |
| 1 | 0 | **1** |
| 1 | 1 | **1** |

- **Derived Logical operators (NAND, NOR, XOR, XNOR)**

The NAND Operation (NOT AND)

The NOR Operation (NOT OR)

The XOR Operation ($A \oplus B = \overline{A}B + A\overline{B}$)

The XNOR Operation (NOT XOR)

### II.3. Logic Functions

Logic functions describe the relationship between binary inputs and a single binary output**.** They form the foundation of digital systems and circuits**,** where every computation or decision is reduced to operations on 0s and 1s**.**

$$F(X_1,X_2,\ldots,X_n):\{0,1\}^n \rightarrow \{0,1\}$$

Takes n binary inputs and produces one binary output.

Logic functions can be represented in three main forms:

1. **Algebraic Expression** (Boolean formula) Example : $F(A,B) = AB + A\bar{B} + \overline{AB}$
2. **Truth Table:** Lists all possible inputs and outputs.
3. **Logic Circuit (Gate Diagram):** Graphical representation using logic gate symbols.

In Boolean algebra**,** logic functions are expressed using algebraic expressions that describe the relationship between binary inputs and outputs. These expressions use Boolean operators (AND, OR, NOT, etc.) to define the function in symbolic form.

Examples of Algebraic Expressions

1. **Simple Functions**
   $F(A,B)=A+B \rightarrow$ *Logical OR*
   $F(A,B)=AB \rightarrow$ *Logical AND*
   $F(A) = \bar{A} \rightarrow$ *Logical NOT*
2. **Composite Functions**
   $F(A,B,C)=(A+B)\cdot C$
3. **Canonical Algebraic Forms**
   Sum of Products (SOP) (OR ( + ) of multiple AND terms)
   $F(A,B,C)=ABC+ABC+ABC$
   Product of Sums (POS) (AND ( · ) of multiple OR terms)
   $F(A,B,C)=(A+B)(A+C)(B+C)$

The algebraic expression representation is a symbolic way to describe logic functions using Boolean operators (AND, OR, NOT, etc.). It is widely used because it connects truth tables and circuit design, while also allowing simplification through Boolean algebra laws.

1. **Inputs:** Each column corresponds to a binary input variable (A, B, C, …).
2. **Outputs:** One or more columns correspond to the logic function(s).
3. **Rows:** Each row represents one possible input combination. For **n** input variables, the truth table has $\mathbf{2^n}$ rows.

**Example with Three Variables**

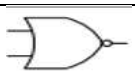
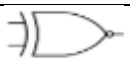
$F(A,B,C) = (A+B).\bar{C}$

| *A* | *B* | *C* | *A+B* | $\bar{C}$ | *F(A,B,C)* |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

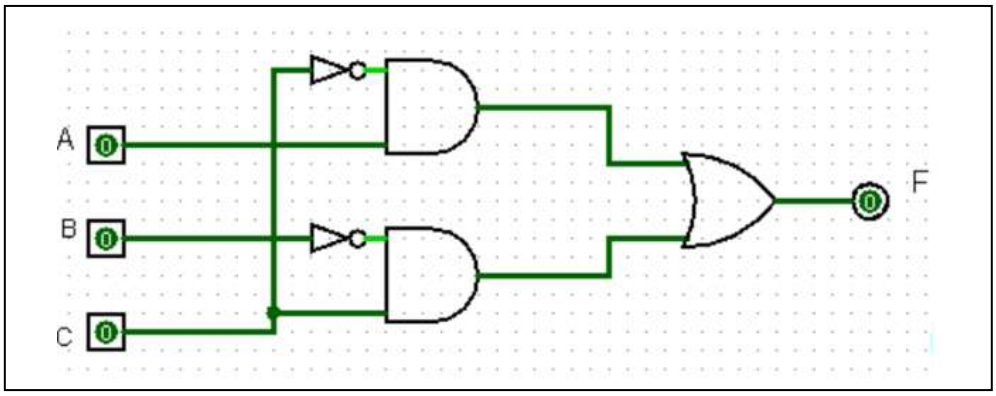
Truth tables provide a systematic and visual method to analyze Boolean functions. They show every possible input combination and the resulting output, making them a fundamental tool in digital logic design.

A logic circuit representation, also known as a gate diagram, is a graphical method of illustrating Boolean expressions or truth tables using logic gate symbols. It visually shows how binary inputs are processed through gates to produce the required output.

Each Boolean operator has a corresponding electronic symbol used in circuit diagrams. The Basic Logic Gate Symbols:

| Gate | Symbol | Boolean Expression | Function | Integrated Circuits (TTL 74xx) | Integrated Circuits (CMOS 40xx) |
|---|---|---|---|---|---|
| **AND** |  | $F = A.B$ | Output = 1 if all inputs = 1 | 7408 | 4081 |
| **OR** |  | $F = A + B$ | Output = 1 if at least one input = 1 | 7432 | 4071 |
| **NOT** |  | $F = \bar{A}$ | Output is the inverse of input | 7404 | 4069 |
| **NAND** |  | $F = \overline{A.B}$ | Inverse of AND | 7400 | 4011 |
| **NOR** |  | $F = \overline{A + B}$ | Inverse of OR | 7402 | 4001 |
| **XOR** |  | $F = A \oplus B$ | Output = 1 if inputs differ | 7486 | 4030 |
| **XNOR** |  | $F = \overline{A \oplus B}$ | Output = 1 if inputs are equal | 74266 | 4077 |

**Example with Function $F(A,B,C) = \bar{B}C + A\bar{C}$**




## III. Simplifying logic functions

The Power of Simplification of Boolean function, that can be directly implemented as a logic circuit, but the expression obtained from a truth table is often not the most efficient. The goal of simplification is to derive an equivalent expression that requires fewer gates and/or fewer inputs per gate. This reduction offers several advantages: lower cost, since fewer chips and components are needed; increased reliability, as fewer components mean fewer potential points of failure; and faster operation, thanks to reduced propagation delay through fewer logic levels.

Simplifying logic functions is a common task in digital design and computer science. The goal is to simplify a Boolean expression or logic circuit while preserving its functionality. There are several techniques for simplifying logic functions, including:

- **Boolean Algebra:** Use Boolean algebra laws and theorems to manipulate and simplify expressions. Common laws include the commutative, associative, distributive, identity, and complement laws.

- **Karnaugh Maps (K-Maps):** It helps simplify Boolean expressions and minimize logic circuits. Karnaugh Maps are particularly useful for functions with a small number of variables.

### III.1. Boolean Algebra method

The **Boolean Algebra method** is one of the most fundamental approaches to simplifying logic functions. It relies on the application of Boolean laws, rules, and theorems to manipulate expressions into simpler, but logically equivalent, forms.

The objective is to reduce the number of logic gates, inputs, and overall complexity of the circuit while preserving its functionality.

Steps in the Boolean Algebra Simplification Method:

1. **Write the expression** from the truth table or problem statement.
2. **Apply Boolean laws and theorems** systematically to reduce terms.
3. **Eliminate redundant variables or terms** whenever possible
4. **Stop when no further simplification** is possible.

**Example: Simplify $F(A, B, C) = AB + A\bar{B}$**

**Step 1**: Factor A : $F = A(B + \bar{B})$
**Step 2**: Apply Complement Law: $B + \bar{B} = 1$
**Step 3**: Simplify: $F = A.1 = A$

The Boolean algebra method is a fundamental skill for anyone working with digital logic. While visual methods like Karnaugh Maps are often faster and guarantee a minimum form, the algebraic method is indispensable for formal verification and understanding the underlying principles of logic simplification.

### III.2. Karnaugh Maps method (K-Maps)

To visually simplify Boolean expressions into their minimal Sum-of-Products (SOP) or Product-of-Sums (POS) form by grouping adjacent cells in a special grid. Boolean algebra simplification requires intuition and practice. The K-map method provides a systematic, visual technique that:

- Guarantees the simplest SOP or POS expression.
- Is much faster and less error-prone than algebraic manipulation for 2-5 variables.
- Makes it easy to identify and eliminate redundant terms.

A K-map is a graphical representation of a truth table. The key innovation is that adjacent cells differ in only one input variable. This layout visually highlights terms that can be combined. **Gray Code Order:** The row and column headers are not in binary order (00, 01, 10, 11) but in Gray code order (00, 01, 11, 10). This ensures adjacent cells are always logically adjacent (differ by one bit).

**Common K-Map Sizes:**

- 2-Variable: 2x2 grid (4 cells)
- 3-Variable: 2x4 grid (8 cells)
- 4-Variable: 4x4 grid (16 cells)

**Example: 3-Variable amd 4-Variable K-Maps Structure**

| A\BC | 0 0 | 0 1 | 1 1 | 1 0 |
|---|---|---|---|---|
| 0 | m0 | m1 | m3 | m2 |
| 1 | m4 | m5 | m7 | m6 |

| AB\CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | m0 | m1 | m3 | m2 |
| 01 | m4 | m5 | m7 | m6 |
| 11 | m12 | m13 | m15 | m14 |
| 10 | m8 | m9 | m11 | m10 |

**Example: Function with the following truth table**

| A | B | C | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

| A\BC | 0 0 | 0 1 | 1 1 | 1 0 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 |

## IV. System Overview

The Digital Electronics practical workshop begins with a general introduction to the principles of digital logic and to the experimental hardware used throughout the laboratory sessions. Before carrying out the different experiments, it is essential for students to become familiar with the basic concepts of Boolean algebra, the representation of logic levels, and the use of the LAB KL-300 Digital Logic Training System.

This introductory session aims to review the theoretical background of digital systems and to provide students with a clear understanding of how logical operations are implemented in real circuits. The LAB KL-300 is a complete and autonomous training platform designed for studying and testing a wide variety of digital logic circuits. It includes thirteen functional

modules (KL33001 – KL33013), each dedicated to a specific type of logic device such as gates, arithmetic units, encoders, decoders, multiplexers, flip-flops, and counters.

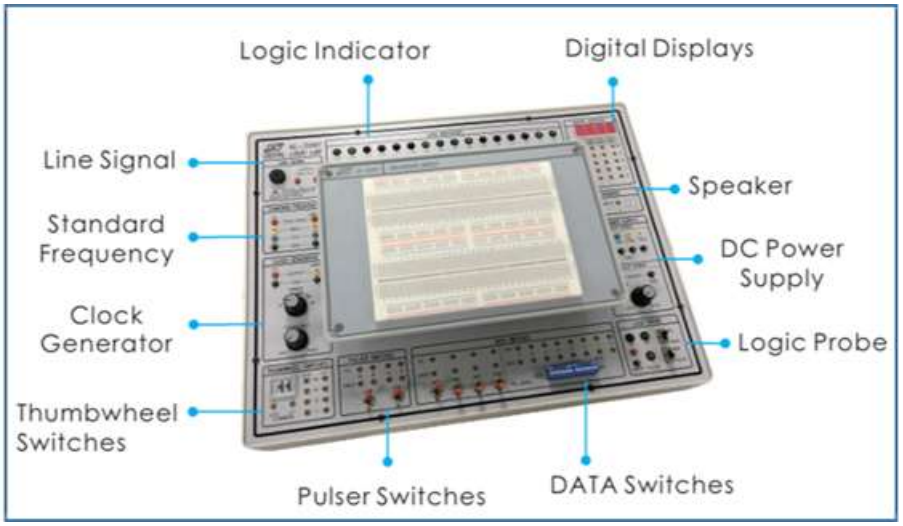



Figure 1: KL-300 Basic Unit with a Breadboard Module

The system is equipped with switches, LEDs, logic indicators, and clock generators, allowing direct visualization of logical states and signal transitions. By connecting the modules with patch cords, students can design and test different logic configurations while observing the resulting output behavior in real time. The platform operates with a standard 5V DC power supply, ensuring safe and stable laboratory conditions.

During this first session, students are expected to identify each module, understand its internal structure and pin configuration, and learn the correct method for wiring and testing circuits on the bench. A preliminary review of truth tables, logic symbols, and Boolean equations will also be conducted to ensure a smooth transition to the subsequent workshops.

At the end of this introduction, students should be able to operate the LAB KL-300 system confidently, interpret the function of each module, and prepare simple logic circuits using fundamental gates. This foundational knowledge will be essential for the successful completion of the following practical sessions, which progressively explore more complex combinational and sequential digital systems.

Here's a clear and professional table summarizing the 13 modules (KL33001 – KL33013) of the LAB KL-300 Digital Logic Training System.

| Module No. | Module Name and Function | Description |
|---|---|---|
| **KL33001** | Basic Logic Gates I | Contains fundamental logic gates such as AND, OR, and NOT; used for simple logic function experiments. |
| **KL33002** | Basic Logic Gates II | Provides additional logic gates including NAND, NOR, XOR, and XNOR; allows verification of Boolean laws and De Morgan's theorem. |
| **KL33003** | Universal Gates | Demonstrates the implementation of any logical function using only NAND or NOR gates. |
| **KL33004** | Half and Full Adder | Used to study binary addition; includes circuits for half-adder and full-adder operations. |
| **KL33005** | Parallel Adder / Subtractor | Demonstrates 4-bit binary addition and subtraction with carry and borrow functions. |
| **KL33006** | Arithmetic Logic Unit (ALU) | Performs arithmetic and logic operations; illustrates how ALUs form part of processor architecture. |
| **KL33007** | Encoder | Converts multiple input signals into coded outputs; used to study data compression and signal coding principles. |
| **KL33008** | Decoder | Converts coded inputs into multiple output signals; essential for address decoding and display applications. |
| **KL33009** | Multiplexer / Demultiplexer | Demonstrates data selection and routing; used to study signal transmission control in digital systems. |
| **KL33010** | RS Flip-Flop | Provides circuits for studying asynchronous and synchronous RS flip-flops; introduces memory behavior. |
| **KL33011** | D Flip-Flop | Demonstrates data storage and clocked signal control; fundamental for registers and memory design. |
| **KL33012** | JK Flip-Flop | Illustrates the operation of JK flip-flops and their use in toggle and counting applications. |
| **KL33013** | Counter and Clock Generator | Used to build and test asynchronous and synchronous binary or decimal counters; includes variable clock sources. |

# Workshop 1: Study and Design of Combinational Basic Logic Gates

## I. Objectives

At the end of this lab, the student should be able to:

- Become familiar with the **KL-300 Digital Logic Laboratory System**;
- Test the operation of the **NOR, NAND, and XOR** gates using the **KL-33001** and **KL-33002** modules;
- Build the **NOT, OR, and AND** gates using only a **NOR** gate;
- Build the **NOT, OR, and AND** gates using only a **NAND** gate;
- Implement an **XOR** gate using different combinations of logic gates (NOT, OR, AND, NAND, NOR), for example with four NAND gates.

## II. Required Materials and Equipment

- KL-300 basic unit;
- Kl-33001 and KL-33002 modules;
- Connecting wires.

## III. KL-300 system Overview

The KL-300 Digital Logic Laboratory is a complete stand-alone system designed for teaching and practical training in digital electronics.

It enables students to:

- Perform real experiments with logic gates and combinational/sequential circuits;
- Move from Boolean algebra theory to the practical implementation of logic functions;
- Develop a deeper understanding through hands-on experimentation, signal manipulation, and result observation.
- System Components:
- A basic unit (Figure 1.1) with a stabilized power supply, function generators, input switches, LED indicators and digital displays.
- A set of 13 interchangeable modules (KL-33001 à KL-33013), each dedicated to a specific topic (logic gates, flip-flops, counters, memories, etc.).

- This modular structure offers flexibility and allows progressive construction, testing, and analysis of logic circuits.

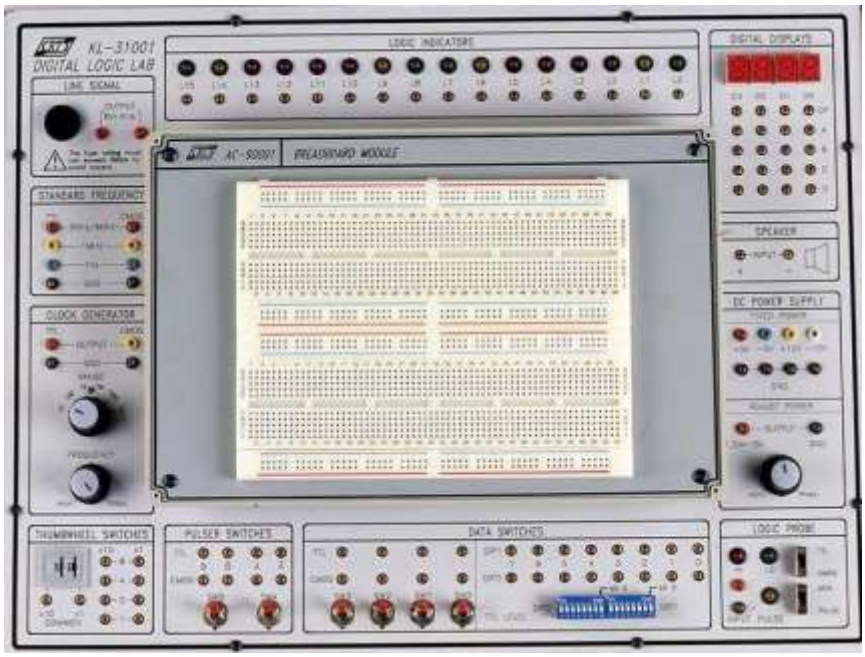



**Figure 1.1.** UKL-300 Basic Unit with a Breadboard Module

## VI. Work requested

The objective of this session is to introduce students to the fundamental building blocks of digital systems, such as AND, OR, NOT, NAND, NOR, XOR, and XNOR gates. Through analysis, truth table construction, and practical circuit design, students will gain a clear understanding of how logical operations are implemented and combined to form more complex digital functions. This workshop serves as a foundation for all upcoming activities in the module.

### VI.1. Implementing NOT, OR and AND Gates with a NOR Gate

- Write the truth table and Boolean expression of the NOR gate.
- Show that AND, NOT, and OR can be derived from the NOR operator.
- Verify experimentally with the KL-300 and KL-33002 (block a).

### VI.1.1. NOR Gate Verification

Test gate U1a (Figure 1.2-a).

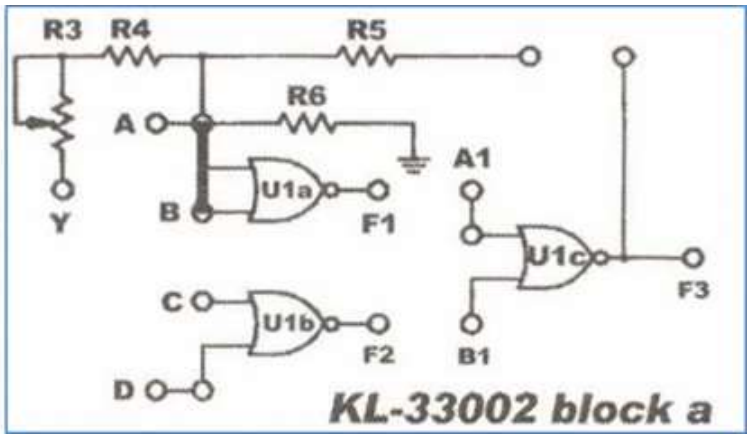



**Figure 1.2-a**

- Connect the **+5V (TTL)** terminal to the +5V (fixed power) output and connect the grounds.
- Connect inputs **A** and **B** to switches **SW0** and **SW1**, and output **F1** to indicator **L1**.
- Vary SW0 and SW1, record output F1, and complete the truth table.

### IV.1.2. NOT Gate Construction

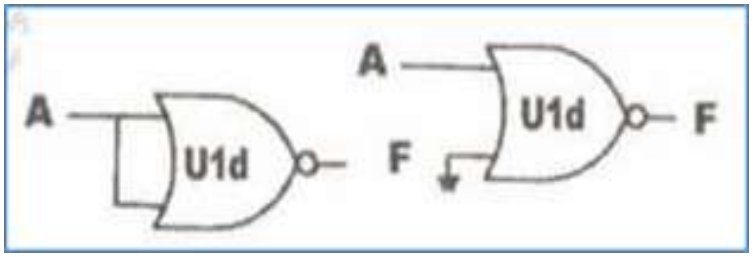



**Figure 1.2-b**

Connect A and B together (figure 1.2-b), test with SW0, and check behavior (justify).

### IV.1.3. OR Gate Construction

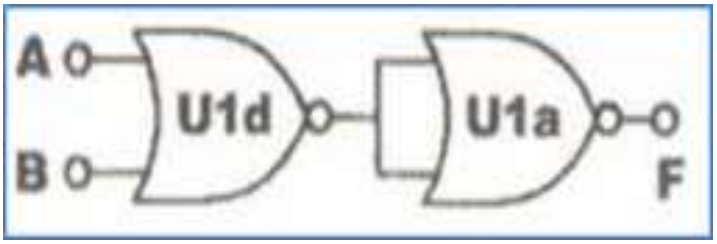



**Figure 1.2-c**

- Verify the truth table at output F3 and confirm if the behavior corresponds to an OR gate.

### IV.1.4. AND Gate Construction

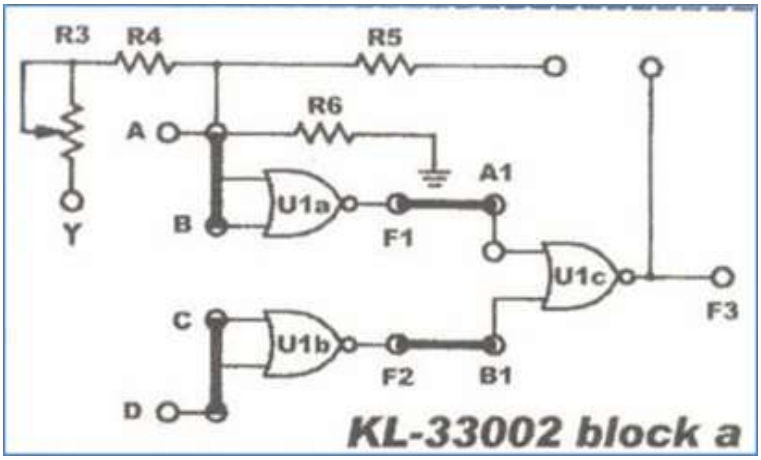



**Figure 1.2-d**

- Verify the truth table at output F3 and check if the behavior corresponds to an AND gate.

### IV.2. Implementation NOT, OR and AND Gates Using a NAND Gate

- Write the **truth table** and **Boolean expression** of the NAND gate.
- Show that **AND, NOT, and OR** can be derived from the **NAND operator**.
- Verify experimentally with **KL-33002 module (block b)**.

### IV.2.1. NAND Gate Verification

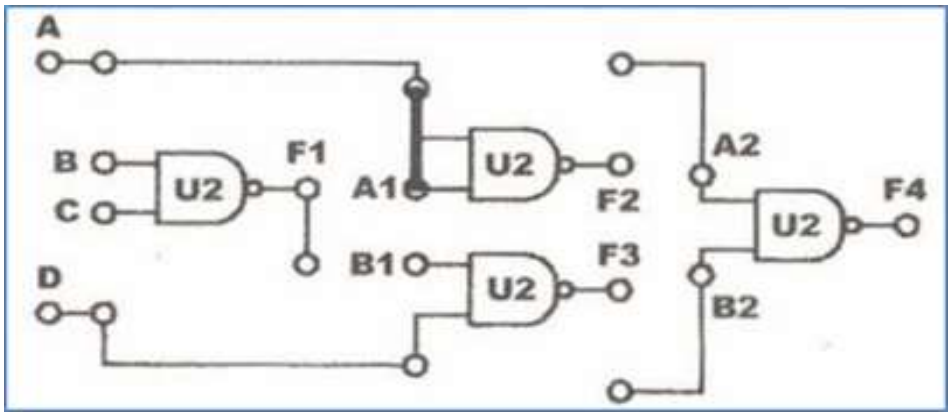



**Figure 1.3-a**

- Connect inputs **B** and **C** to switches SW0 and SW1, and observe **F2**.
- Complete the truth table.

### IV.2.2. NOT Gate Construction

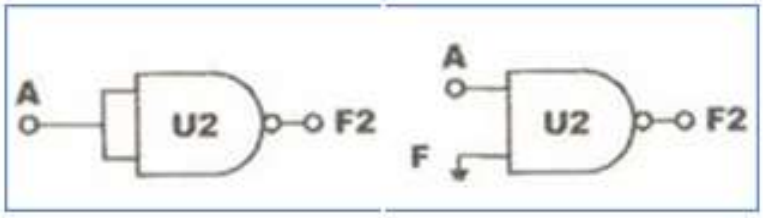



**Figure 1.3-b**

- Connect both inputs together and link to SW0 (figure 1.3-b).
- Connect the output to L1.
- Complete the truth table and check if the behavior corresponds to a NOT gate (justify).

### IV.2.3. OR Gate Construction

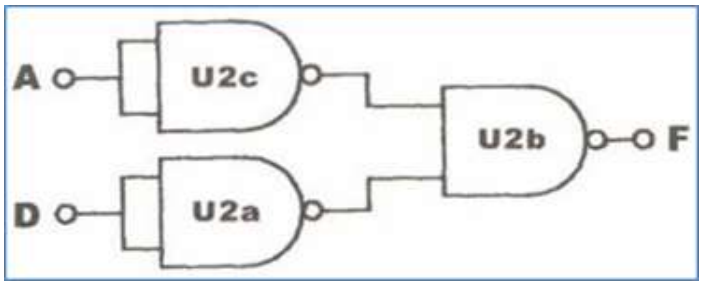



**Figure 1.3-c**

- Verify output **F4** according to inputs SW0 and SW1.
- Confirm if the circuit behaves as an OR gate (justify).

### IV.3. Implementation of an XOR Gate Using NAND Gates

- Write the **truth table** and **Boolean expression** of the XOR gate.
- Show that the XOR function can be implemented using **NAND gates**.
- Build an XOR gate with U2a, U2b, U2c, U2d as shown in Figure 1.4.

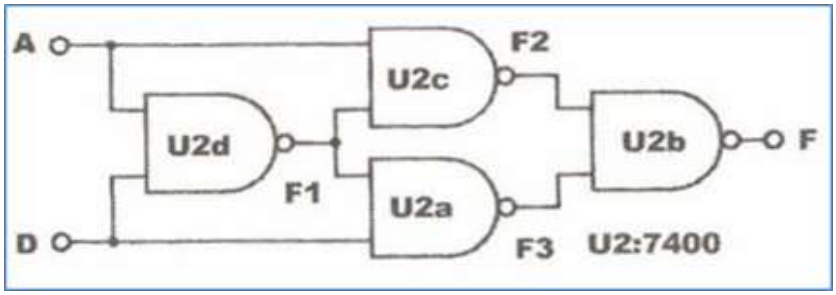



**Figure1.4.** Logic Gate Circuit Using 4 NAND Gates

- connect inputs SW0 and SW1, observe F4, and verify operation(justify).

## V. Conclusion

Provide a conclusion that highlights the objectives that were achieved, the main steps that were completed, and the key lessons gained from this lab session.

Through this first workshop, students were able to explore and manipulate the fundamental logic gates that form the basis of all digital electronic systems. Starting from the **NOR** and **NAND** gates, they demonstrated how any logical function—**NOT, OR, AND, and XOR**—can be derived and implemented using only one universal gate. By constructing truth tables, analyzing Boolean expressions, and validating results through experimental testing on the **KL-300/KL-33002 modules**, students reinforced the link between theoretical logic concepts and their practical hardware implementation. This session established a solid foundation for the understanding and design of more complex digital circuits that will be studied in the following workshops.

# Workshop 2: Combinational Circuits – Arithmetic Units

## I. Objectives

The objective of this workshop is to introduce students to the design and synthesis of arithmetic combinational circuits such as half adders, full adders, and binary adders. Through analysis and practical implementation, students will learn how to represent arithmetic operations using logic gates and how to construct functional circuits capable of performing binary addition. This session aims to strengthen the understanding of digital arithmetic and prepare students for more advanced combinational designs.

At the end of this lab, the student should be able to:

- Implement a half-adders and a full adder;
- Implement a half-subtractors and a full subtractors;
- Design a 4-bit binary adder/subtractor using logic circuits.

## II. Required Materials and Equipment

To carry out this workshop, the following materials and laboratory equipment are required:

- KL-300 basic unit;
- Kl-33004 module;
- Connecting wires.
- Switches (SW0, SW1, etc.) for binary input selection
- LED indicators or display outputs for result visualization
- +5V DC power supply (TTL compatible)

## III. Theoretical Background

### III.1. The Adder

The Arithmetic Logic Unit (ALU) is an essential part of a computer that performs calculations. It is integrated into the Central Processing Unit (CPU) or microprocessor and is made up of logic gate circuits.

One of the main functions of the ALU is addition. In this experiment, we will design the circuit that performs this arithmetic operation.

#### III.1.1. Half-Adder

A half-adder is a combinational circuit that computes the arithmetic sum of two one-bit inputs A and B, without considering any carry from a previous stage.

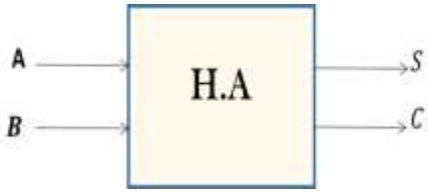



**Figure 2.1.** Block diagram of a half-adder

### III.1.2. Full-Adder

A full-adder is a combinational circuit that computes the sum of two bits along with the carry from the previous stage. The principle is the same as in decimal arithmetic.

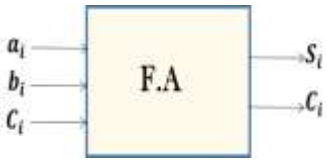



**Figure 2.2.** Block diagram of a full-adder

To perform multi-bit addition, several full-adders can be cascaded. This arrangement enables parallel computation of sums, but each stage must wait for the carry from the previous stage before completing its operation.

For example, the sum S2 of adder Add2 can only be computed once the carry from adder Add1 has been determined.

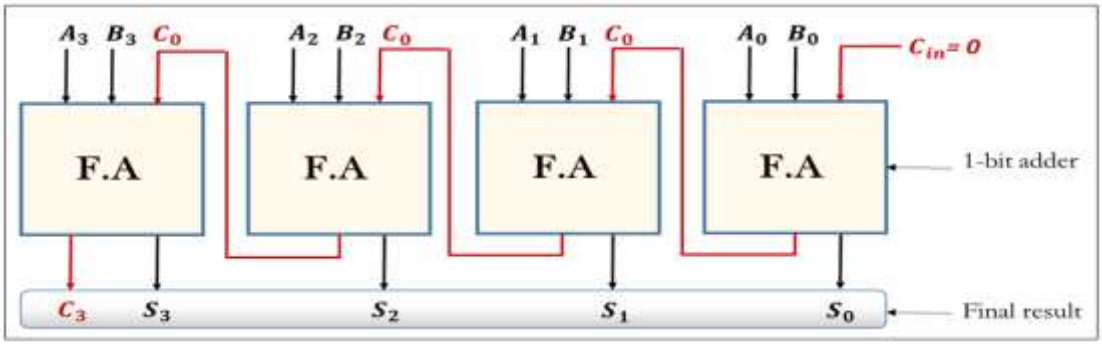



**Figure 2.3.** Block diagram of a 4-bit parallel adder

## III.2. The Subtractor

A subtractor is a combinational logic circuit designed to perform binary subtraction between two bits or binary numbers. Similar to adders, subtractors can be implemented in two forms: Half Subtractor and Full Subtractor.

### III.2.1. Half-Subtractor

Est A half-subtractor is a logic circuit used in digital systems to subtract two 1-bit binary numbers. It has two inputs and two outputs: the difference and the borrow.

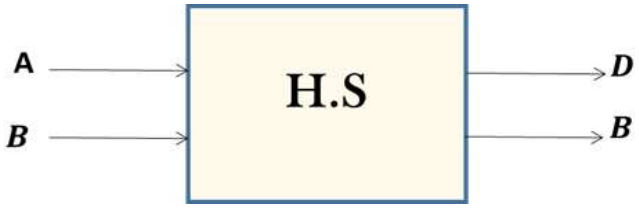



**Figure 2.4.** Block diagram of a half-subtractor

### III.2.2. Full-Subtractor

A full-subtractor has three inputs and two outputs. It performs subtraction while also accounting for the borrow from the previous stage.

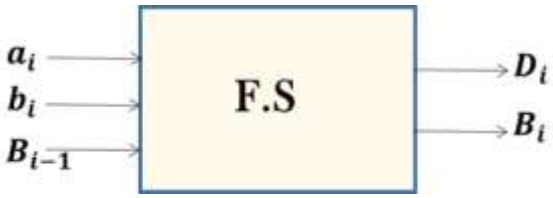



**Figure 2.5.** Block diagram of a full-subtractor

### III.3. Subtractor Using an Adder

A negative integer can be expressed in two's complement form:

$$A - B = A + (-B) = A + \overline{B} + 1$$

Thus, subtraction can be implemented using an adder. In practice, the constant « 1 » is generated by applying +5V, and $\overline{B}$ is obtained using an inverter.

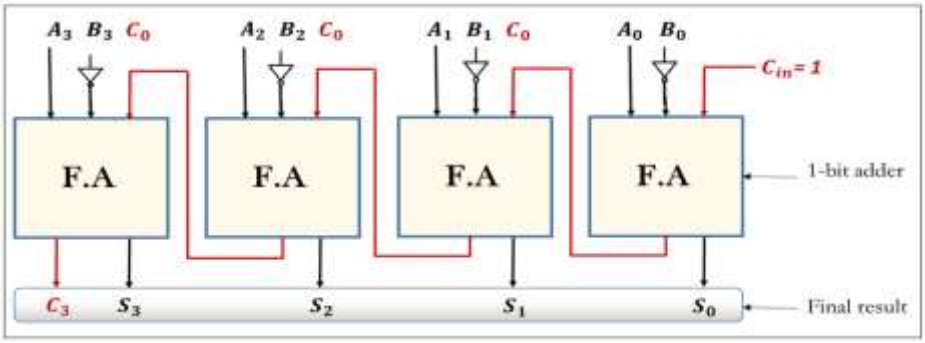



**Figure 2.6.** Block diagram of a 4-bit subtractor using full-adders

## IV. Work requested

This activity introduces the concept of binary subtraction in digital systems through the design and implementation of Half Subtractor and Full Subtractor circuits. Using the KL-33004 training module (see Figure 2.6-a and Figure 2.6-b), students will analyze how subtraction can be achieved with logic gates and how the Difference and Borrow signals are generated. By constructing truth tables and experimentally verifying the behavior of the circuits, students will strengthen their understanding of arithmetic logic design and gain practical skills in implementing digital subtraction using hardware components.

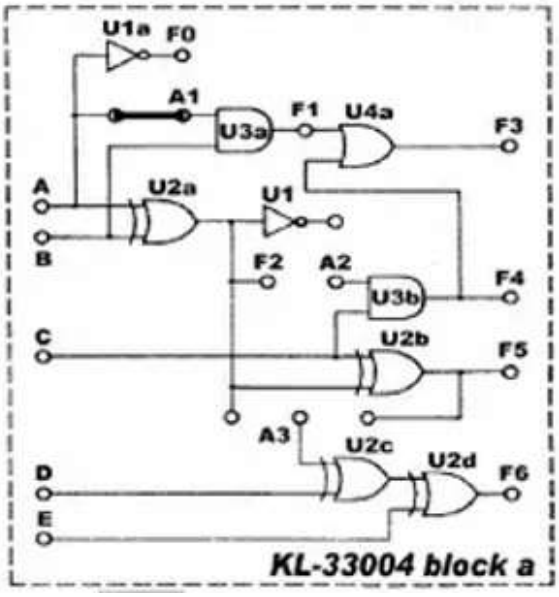



**Figure 2.6-a.** Block a from KL-33004 module.

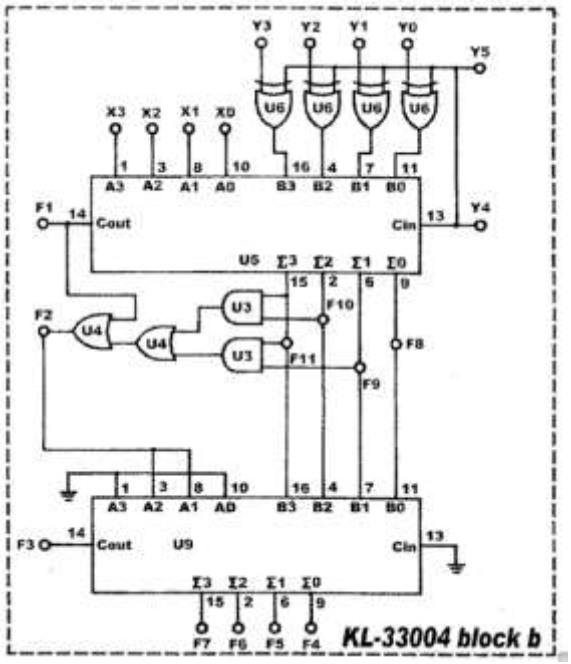



**Figure 2.6-b.** Block b from KL-33004 module.

### IV.1. Half-Adder

- Explain in detail the operating principle of a half-adder (truth table and simplification), then draw the logic circuit.
- Using module KL-33004, build the half-adder and complete the connections with block a **(Figure 2.6-a)**.
- Connect the +5V (TTL) terminal of the module to the +5V output of the power supply and tie the grounds together. Connect inputs A and B to TTL outputs of switches SW0 and SW1 (Data Switch), and outputs F1 and F2 to the logic indicators.

- Toggle switches SW0 (A) and SW1 (B) between 0 and 1, observe outputs F1 (carry) and F2 (sum), and fill in the truth table below.

| **Inputs** | | **Outputs** | |
|---|---|---|---|
| SW0(A) | SW1(B) | F2(S) | F1(C ) |
| | | | |
| | | | |
| | | | |
| | | | |

### IV.2. Full-Adder

- Explain in detail the operating principle of a full-adder (truth table and simplification), then draw the logic circuit.
- Using module KL-33004, build the full-adder and complete the connections with block a **(Figure 6)**.
- Connect inputs A, B, and Cin to TTL outputs of switches SW0, SW1, and SW2, and connect the outputs to the logic indicators.
- Toggle switches SW0(A), SW1(B), and SW2(Cin) between 0 and 1, observe outputs S (sum) and C (carry), and complete the table below.

| **Inputs** | | | **Outputs** | |
|---|---|---|---|---|
| SW0(A) | SW1(B) | SW2($C_{-1}$ ) | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

### IV.3. Half-Subtractor

- Explain in detail the operating principle of a half-subtractor (truth table and simplification), then draw the logic circuit.
- Compare it with the half-adder and explain the differences.

- Using module KL-33004, build the half-subtractor and complete the connections with block a **(Figure 2.6-a)**.
- Connect inputs A and B to TTL outputs of switches SW0 and SW1, and outputs F1 and F2 to the logic indicators.
- Toggle switches SW0(A) and SW1(B) between 0 and 1, observe outputs F1 (borrow) and F2 (difference), and fill in the table below.

| **Inputs** | | **Outputs** | |
|---|---|---|---|
| SW0(A) | SW1(B) | F2(S) | F1(C ) |
| | | | |
| | | | |
| | | | |
| | | | |

### IV.4. Full-Subtractor

- Explain in detail the operating principle of a full-subtractor (truth table and simplification), then draw the logic circuit.
- Using module KL-33004, build the full-subtractor and complete the necessary connections with block a (Figure 2.6-a).
- Connect inputs A, B, and Cin to TTL outputs of switches SW0, SW1, and SW2, and connect outputs to the logic indicators.
- Toggle switches SW0(A), SW1(B), and SW2(Cin) between 0 and 1, observe outputs S (difference) and C (borrow), and complete the table below.

| **Inputs** | | | **Outputs** | |
|---|---|---|---|---|
| SW0(A) | SW1(B) | SW3 ($C_{-1}$) | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

### IV.5. The 4-bit Parallel Adder

- Explain in detail the operating principle of a 4-bit adder (truth table and simplification).
- Using module KL-33004, build the 4-bit adder (use U5 as a 4-bit adder) and complete the connections with block b **(Figure 2.6-b)**.
- Connect inputs X0–X3 and Y0–Y3 to Data Switches DIP1.0–DIP1.3 and DIP2.0–DIP2.3, respectively. Connect outputs F1 (Cout), F8, F9, F10, and F11 to logic indicators. Set Y5 (SW0) = 0 to perform addition, and Y4 or Cin (SW1) = 0.
- Toggle the DIP switches, record the outputs in both binary and hexadecimal, and complete the table.

| **Inputs** | | **Outputs** | | |
|---|---|---|---|---|
| X(hex) | Y(hex) | F(S) (bin) | F(S) (hex) | Cout |
| 0 | 0 | | | |
| 0 | 1 | | | |
| 1 | 1 | | | |
| 2 | 4 | | | |
| 5 | 5 | | | |
| 6 | 3 | | | |
| 7 | A | | | |
| A | 9 | | | |

### IV.6. The 4-bit Parallel Subtractor

- Explain in detail the operating principle of a 4-bit subtractor (truth table and simplification).
- Using module KL-33004, build the 4-bit subtractor (use U5) and complete the connections with blockb **(Figure 2.6-b)**.
- Connect inputs X0–X3 and Y0–Y3 to Data Switches DIP1.0–DIP1.3 and DIP2.0–DIP2.3, respectively. Connect outputs F1 (Cout), F8, F9, F10, and F11 to logic indicators.
- To perform subtraction, set Y5 (SW0) = 1 (to generate the 1’s complement of Y) and Y4 (SW1) = 1 (to add 1, converting Y to its 2’s complement).
- Toggle the DIP switches, record the outputs in both binary and hexadecimal, and complete the table.

| **Inputs** | | **Outputs** | | |
|---|---|---|---|---|
| X(hex) | Y(hex) | F(S) (bin) | F(S) (hex) | Cout |
| 0 | 0 | | | |
| 0 | 1 | | | |
| 1 | 3 | | | |
| 2 | 4 | | | |
| 2 | 5 | | | |
| 6 | 3 | | | |
| 7 | 2 | | | |
| A | 8 | | | |
| 9 | D | | | |
| A | E | | | |
| F | F | | | |

## V. Conclusion

Provide a conclusion that highlights the objectives that were achieved, the main steps that were completed, and the key lessons gained from this lab session.

Through this exercise, students successfully implemented and tested both Half Subtractor and Full Subtractor circuits using the KL-33004 module. The comparison between theoretical truth tables and experimental results confirmed the correct operation of subtraction logic using basic gates. This practical verification enhanced the students' ability to relate Boolean expressions to physical circuit behavior. It reinforced their foundation in digital arithmetic circuit synthesis, preparing them for more complex combinational logic applications in upcoming workshops.

# Workshop 3: Combinational Circuits – Encoders, Decoders, and Multiplexers

## I. Objectives

The objective of this workshop is to introduce students to the operation and synthesis of encoders, decoders, and multiplexers as essential data routing and selection circuits in digital systems. Through analysis and practical implementation, students will learn how to:

- Understand and implement the wiring of an encoder and a decoder;
- Understand and implement the wiring of a multiplexer;
- Experimentally verify their operation using the KL-33005 and KL-33006 modules.

## II. Required Materials and Equipment

- KL-300 basic unit;
- KL-33005 and KL-33006 modules.
- Input switches (SW0, SW1, SW2, ...) for data and select line configuration
- LED indicators or output display modules for visualizing results
- +5V DC power supply (TTL compatible)
- Connection wires and jumper cables

## III. Theoretical Background

Combinational circuits such as encoders, decoders, and multiplexers play a crucial role in data selection, address translation, and information routing within digital systems.

### III.1. Encoder

An **encoder** is a logic circuit with $2^N$ input lines, only one of which is active at a time, and N output lines, as shown in Figure 3.1.

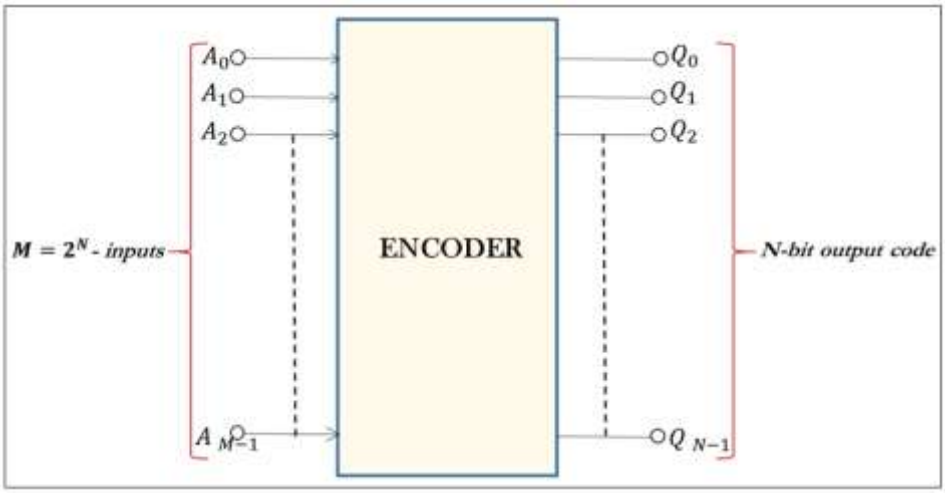



**Figure 3.1.** Block diagram of an encoder

### III.2. Decoder

A decoder is a digital circuit with N inputs and $2^N$ outputs. For each possible input combination, only one output line is activated. Decoders often include one or more enable inputs (E) to control their operation. Figure 3.2 shows the block diagram of an N-bit decoder.

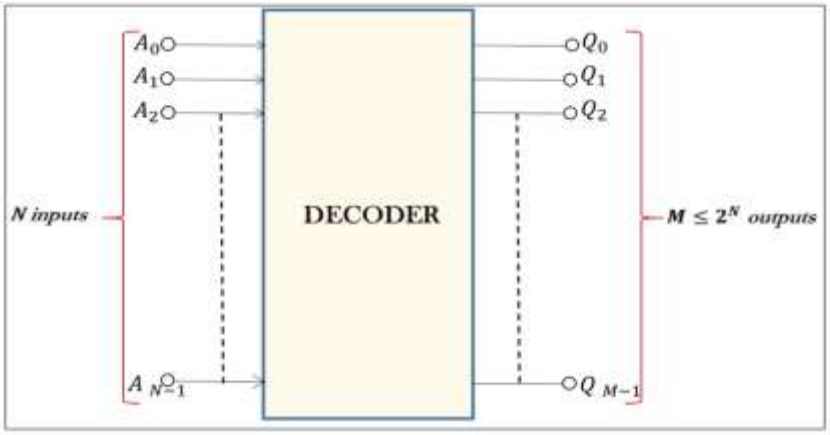



**Figure 3.2.** Block diagram of a Decoder

#### III.2.1. BCD to 7-Segment Decoder

The goal is to display digits from 0 to 9 and letters A to F on a 7-segment display (Figure 3.3). To achieve this, we use a decoder with 4-bit binary input (from 0000 (0) to 1111 (F)) and 7 output signals, each driving one segment of the display. Inputs are labeled E1 to E4 (E1 = LSB). Outputs are Sa, Sb, Sc, Sd, Se, Sf, Sg, connected to the display segments a to g.

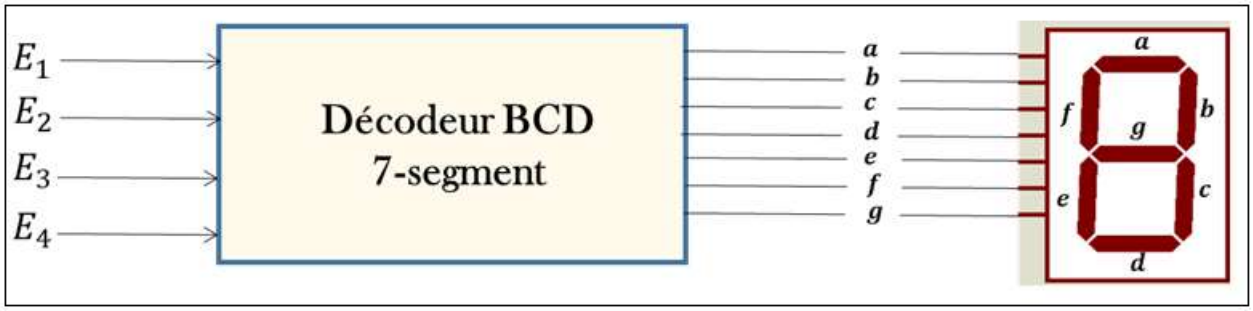



**Figure 3.3.** Block diagram of a BCD to 7-segment decoder

### III.3. Multiplexer

A multiplexer (MUX) is a circuit that selects one data input among several ($D_j$) according to control inputs (addresses $A_i$), and copies its value to the output **Y**. If there are n control inputs, the multiplexer can select among $2^n$ data lines.

In practice, multiplexers are widely used in data transmission systems, where multiple data streams are sent alternately through a single communication line.

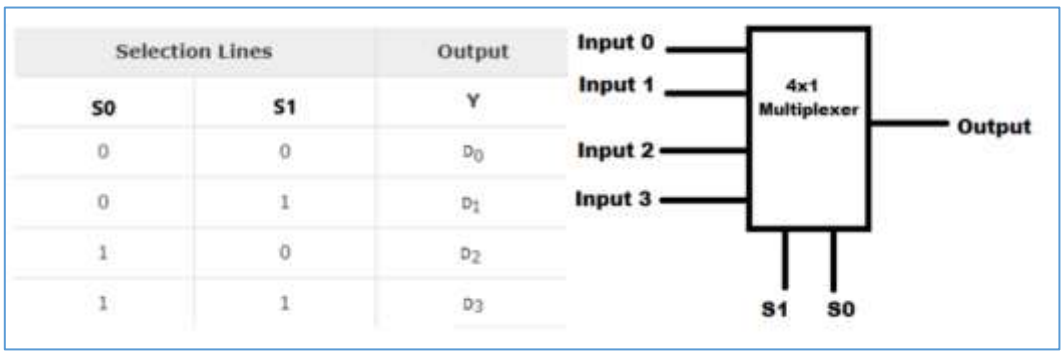



**Figure 3.4.** Block diagram of a 4-to-1 multiplexer

## IV. Work requested

This activity introduces the concept of data encoding, decoding, and selection in digital systems through the design and implementation of Encoder, Decoder, and Multiplexer circuits. Using the KL-33005 and KL-33006 training modules, students will analyze how information can be coded, translated, and routed through combinational logic. By constructing truth tables and experimentally verifying the operation of each circuit, students will observe how select lines, enable inputs, and active logic levels control data flow in digital architectures. This hands-on implementation allows students to strengthen their understanding of combinational data routing logic and develop practical skills in configuring and testing hardware-based encoders, decoders, and multiplexers.

## IV.1. Encoder

### IV.1.1. The 4-to-2 Encoder

- Explain in detail the principle of a 4-to-2 encoder (truth table, logic simplification, logic diagram).
- Implement the circuit using module **KL-33005**, block **a** (Figure 3.5).
- Connect the +5V (TTL) supply and ground. Connect the inputs to the TTL outputs of the **data switches** and the outputs to the logic indicators.
- Vary the switches for inputs A, B, C, and D, record the outputs in a table.

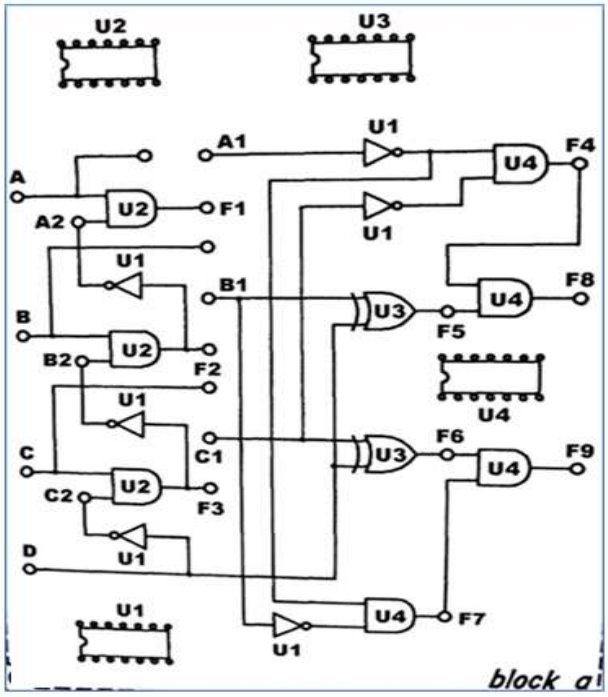



**Figure 3.5. KL33005 block a**

### IV.1.2. Decimal-to-BCD Priority Encoder (10-to-4) – IC 74147

- Complete the truth table of the 74147 priority encoder.
- Implement the 10-to-4 encoder using KL-33006, block **a** (Figure 3.6).
- Connect inputs to DIP switches (DIP1 and DIP0), and outputs F1–F4 to logic indicators or a BCD display.
- Record the output states in a table.

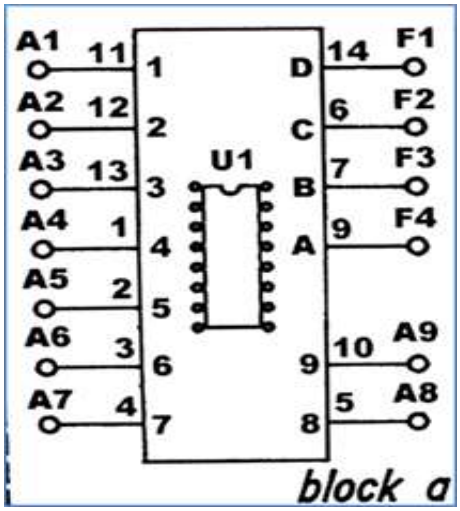



**Figure 3.6.** KL33006 block a

## IV.2. Decoder

### IV.2.1. The 2-to-4 Decoder

- Study a binary-octal decoder (3-to-8)**:** truth table, Boolean expressions, and logic diagram.
- Explain in detail the operation of a 2-to-4 decoder (truth table, logic simplification, logic diagram).
- Implement the circuit using KL-33005**,** block c (Figure 3.7)**.**
- Connect inputs to switches SW0 (A) and SW1 (B), and outputs to logic indicators or a BCD display.
- Vary the switches, record results in a table.

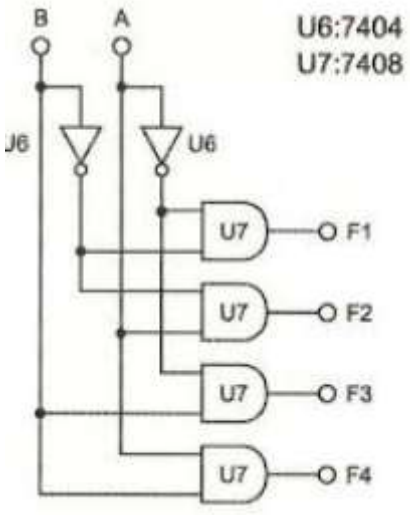



**Figure 3.7.** KL33006 block c

### IV.2.2. The BDC to 7-Segment Decoder

- Complete the truth table of the decoder, then derive simplified equations for the 7 outputs (Sa..Sg) using Karnaugh maps.
- Implement the decoder using **KL-33005**, block **b**, and its 7-segment display (Figure 7).
- Connect inputs A, B, C, and D (decoder U5) to DIP switches, outputs to display DP1, RBI to SW0, and LT to SW1.
- Set RBI = 1 and LT = 1, vary input switches, and observe the display. Record results in a table.

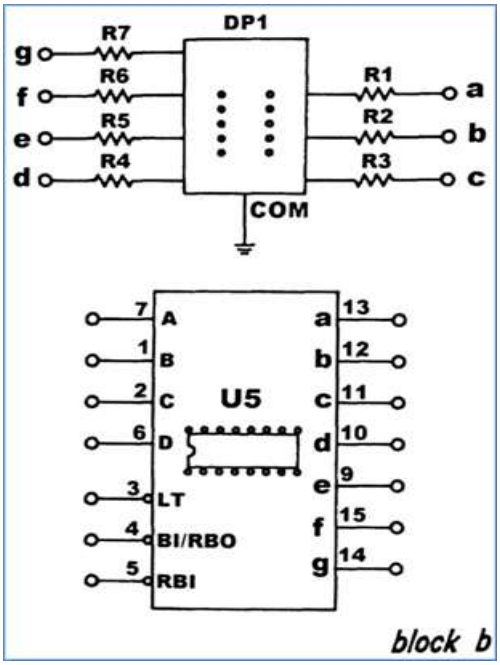



**Figure 7.** KL33005 block b

| *E4* | *E3* | *E2* | *E1* | *Sa* | *Sb* | *Sc* | *Sd* *Se* | *Sf* *Sg* | *Displayed symbol* |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | | 0 |
| 0 | 0 | 0 | 1 | | | | | | 1 |
| 0 | 0 | 1 | 0 | | | | | | 2 |
| 0 | 0 | 1 | 1 | | | | | | 3 |
| 0 | 1 | 0 | 0 | | | | | | 4 |
| 0 | 1 | 0 | 1 | | | | | | 5 |
| 0 | 1 | 1 | 0 | | | | | | 6 |
| 0 | 1 | 1 | 1 | | | | | | 7 |
| 1 | 0 | 0 | 0 | | | | | | 8 |
| 1 | 0 | 0 | 1 | | | | | | 9 |
| 1 | 0 | 1 | 0 | | | | | | A |
| 1 | 0 | 1 | 1 | | | | | | B |
| 1 | 1 | 0 | 0 | | | | | | C |
| 1 | 1 | 0 | 1 | | | | | | D |
| 1 | 1 | 1 | 0 | | | | | | E |
| 1 | 1 | 1 | 1 | | | | | | F |

## IV.3. Multiplexer

### IV.3.1. The 2-to-1 Multiplexer

- Study the 2-to-1 multiplexer: truth table, Boolean expression, logic diagram.
- Implement the circuit using KL-33006, block e, (Figure 8)**.**
- Connect inputs to DIP switches and outputs to logic indicators.
- Observe results and record them in a table.

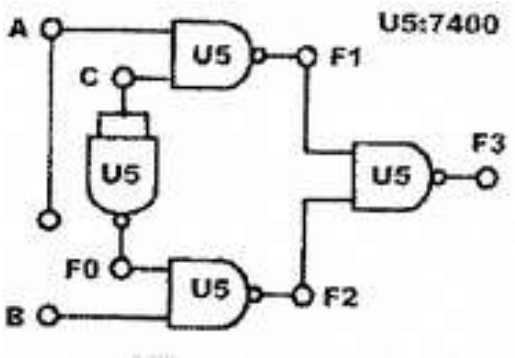



**Figure 8.** KL33006 block e

### IV.3.2. The 8-to-1 Multiplexer – IC 74151

- Complete the truth table of the 8-to-1 multiplexer and derive the output logic equation.
- Implement the circuit using KL-33006, block **f** (IC 74151) (Figure 9).
- Connect inputs to DIP switches, outputs to logic indicators.
- Vary the inputs, observe results, and complete a table.
- Provide a conclusion.

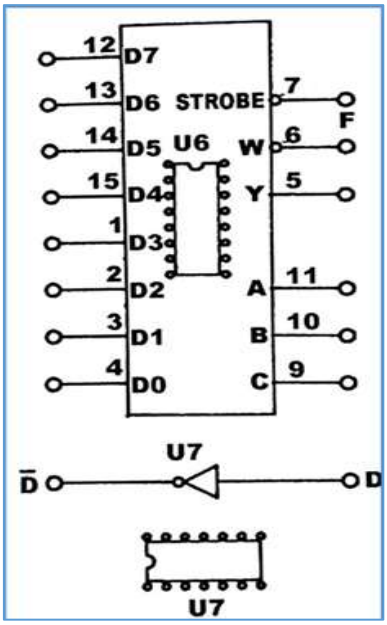



**Figure 9.** KL33005 block f

## V. Conclusion

Write a conclusion summarizing the objectives achieved, the main experimental steps completed, and the lessons learned during this lab.

Through this workshop, students successfully explored the practical implementation of encoders, decoders, and multiplexers using the KL-33005 and KL-33006 modules. By comparing theoretical truth tables with experimental results, they confirmed how data can be encoded, translated, and routed through combinational logic circuits. The manipulation of select lines, input configurations, and output monitoring allowed students to clearly observe the functional differences and relationships between these circuits. This activity reinforced their understanding of data flow control in digital systems and established a solid foundation for more advanced applications in communication buses, memory addressing, and digital system architecture.

# Workshop 4: Sequential Circuits – Flip-Flops (RS, D, JK)

## I. Objectives

The objective of this workshop is to introduce students to **sequential logic circuits** through the study of basic flip-flop elements. At the end of this session, students should be able to :

- Study and test asynchronous and synchronous flip-flops (RS, D, JK).
- Verify truth tables and analyze the effect of inputs and clock signals.
- Gain hands-on experience in building and wiring flip-flop circuits with the KL-300 system and KL-33008 module.

## II. Required Equipment

- KL-300 basic unit;
- Kl-33008 module;
- Clock signal generator (integrated in the KL-300 LAB)
- Input switches (SW0, SW1, etc.) for setting logic levels
- LED indicators for monitoring output states (Q and Not Q)
- +5V DC power supply (TTL compatible)
- Jumper wires and connection cables

## III. Theoretical Background

### III.1. Fundamentals of Flip-Flops

A flip-flop is the most basic sequential circuit, primarily used as a memory element. Its role is to capture temporary information and preserve the state even after the input signal disappears. Since flip-flops operate based on timing signals, time is a critical factor in their behavior.

Flip-flops are bistable devices, meaning they can remain in one of two stable states (0 or 1). For this reason, they serve as elementary memory units. Their state can be changed through one or more control inputs, making them the fundamental building blocks of counters and many other sequential systems.

#### III.1.1. Asynchronous RS Flip-Flop

An asynchronous RS flip-flop operates such that its inputs directly affect the output without waiting for a clock signal. A typical design can be implemented using NAND gates:

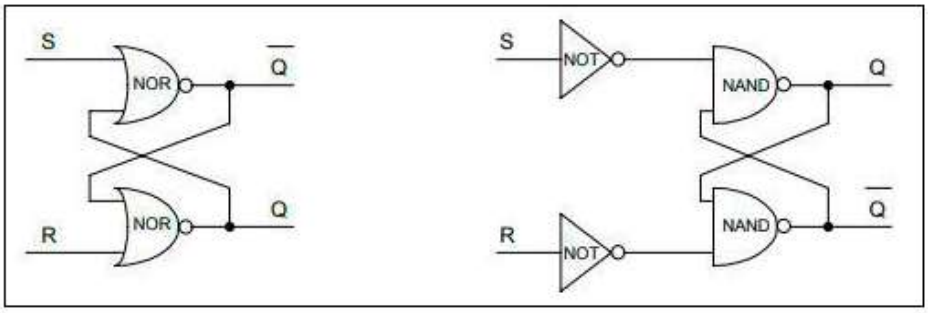



**Figure 4.1.** Asynchronous RS Flip-Flop using NOR or NAND gates

### III.1.2. Synchronous RS Flip-Flop

In contrast, a synchronous RS flip-flop (RSH) only changes its state in response to synchronization pulses, commonly referred to as clock signals.

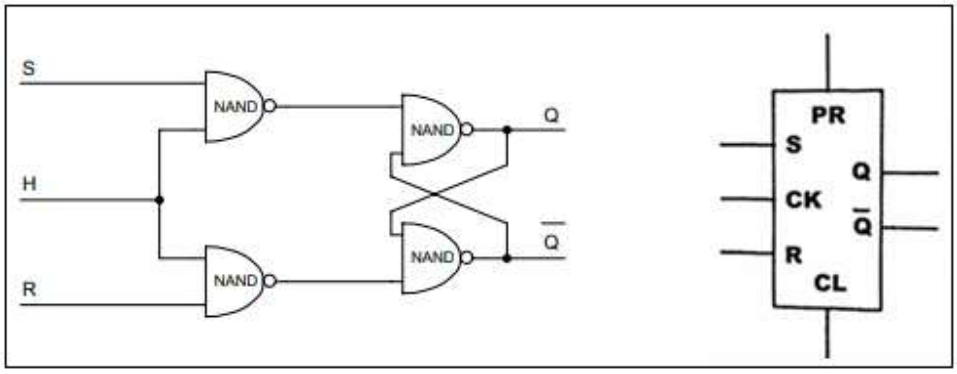



**Figure 4.2.** Synchronous RS Flip-Flop (RSH)

### III.2. D Flip-Flop

A D flip-flop can be built from an RS flip-flop by connecting its inputs through an inverter. This forces the inputs to always take complementary values.

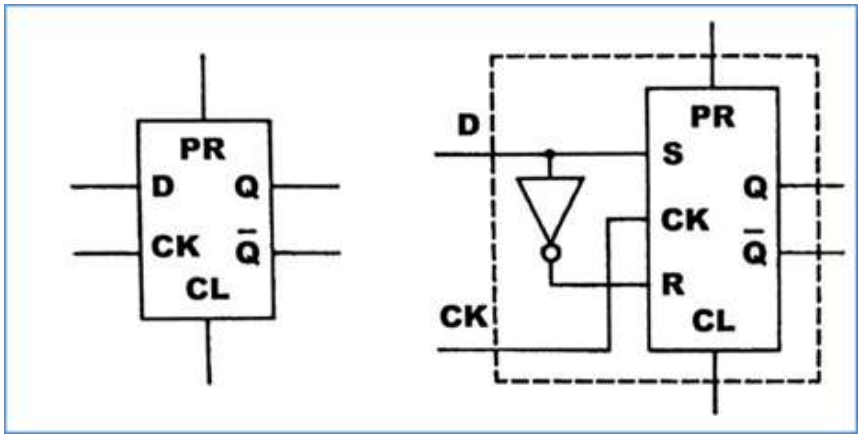



**Figure 4.3.** D Flip-Flop based on RSH

### III.3. JK Flip-Flop

A synchronous JK flip-flop is derived from an RSH flip-flop, with feedback loops from the outputs to the inputs. This configuration removes the undefined state of the RS flip-flop.

The JK flip-flop is the most advanced version and is essential for counting operations. It is implemented as a master-slave RS flip-flop with cross-feedback between outputs and inputs. Using its J and K inputs, it allows counting operations and the ability to preset or stop the counting sequence.

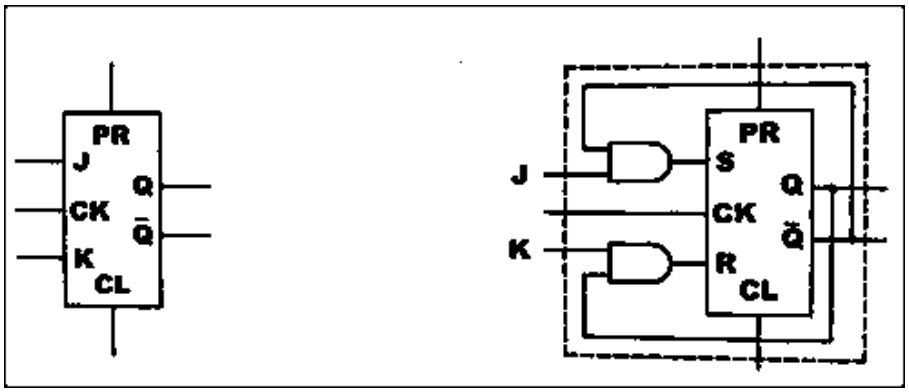



**Figure 4.4.** JK Flip-Flop based on RSH

### IV. Work requested

This activity introduces the concept of sequential logic in digital systems through the design and implementation of RS, D, and JK Flip-Flop circuits. Using the KL-33008 (Block d) training module shown in Figure 4.5, students will analyze how binary information can be stored, synchronized, and controlled using clock-driven bistable elements. By constructing timing tables and experimentally verifying the behavior of each flip-flop type, students will observe how clock pulses, set/reset inputs, and triggering modes influence data transitions in sequential architectures. This practical implementation allows students to strengthen their understanding of memory elements in digital logic and develop hands-on skills in configuring and testing hardware-based flip-flops for reliable state-controlled operations.

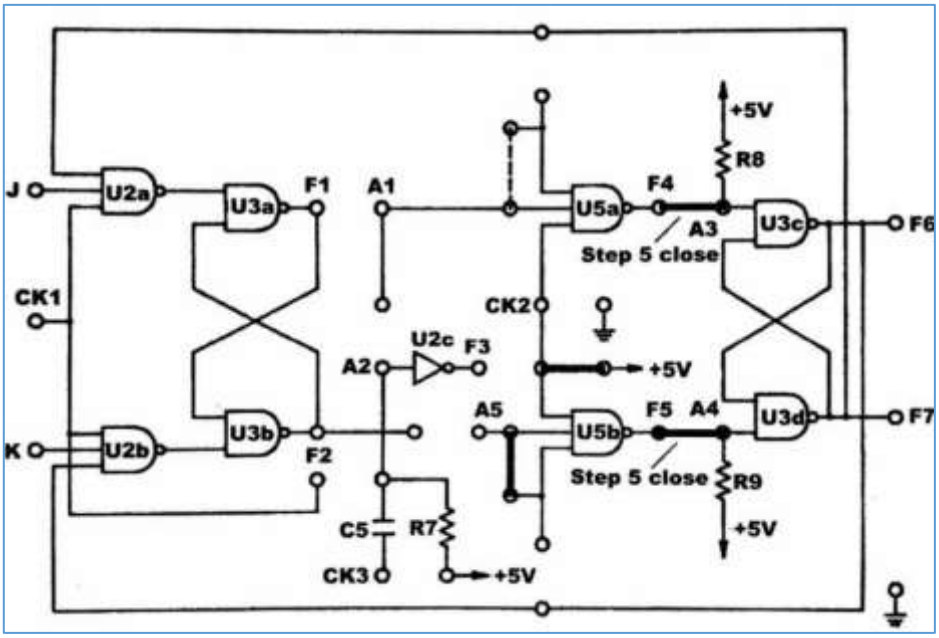



**Figure 4.5.** Block d/KL-33008 Module

### IV.1. Implementation of an Asynchronous RS Flip-Flop

- Recall the algebraic expression and the truth table of the RS flip-flop, then draw the circuit using NOR gates.
- Build the circuit using KL-33008, complete the connections on Block d, and insert jumpers and wires as shown in Figure 4.5-a.
- Connect the +5V (TTL) terminal to the fixed +5V supply, and connect the grounds. Then connect inputs A1 and A5 to the TTL outputs of the Data Switch, and outputs F6 and F7 to logic indicators.
- Apply input sequences and record the corresponding output states in a table.
- Write a conclusion: What is the role of inputs A1 and A5?

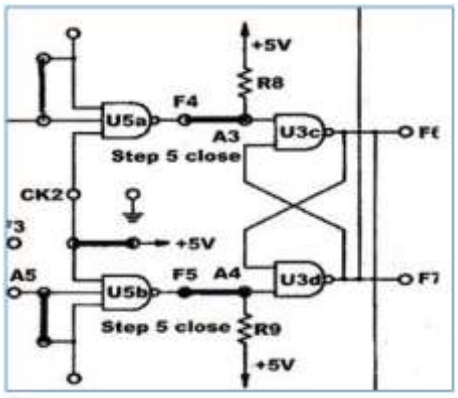



**Figure 4.5-a.** Asynchronous RS Flip-Flop using KL33008 block d

### IV.2. Implementation of a Synchronous RS Flip-Flop

- Use KL-33008 (Block d) to build the circuit, following Figure 4.5-a, but remove the jumper between CK2 and +5V.
- Connect the +5V (TTL) to the power supply and ground. Connect inputs A1 and A2 to the Data Switch outputs, CK2 to the pulser switch output, and outputs F6 and F7 to logic indicators.
- Apply input sequences and record the outputs in a table. Conclusion: What is the role of inputs A1, A2, and the manually controlled clock signal CK2?

### IV.3. Implementation of a D Flip-Flop Based on RS

- Recall the truth table of the D flip-flop, then draw its logic diagram based on RS flip-flop.
- Build the circuit on KL-33008 using Block d, and wire as in Figure 4.5-b.
- Connect input A1 to the Data Switch output, CK2 to the pulser switch output, and outputs F6 and F7 to logic indicators.
- Apply input sequences (A1 and manual CK2 clock) and record the output states in a table.

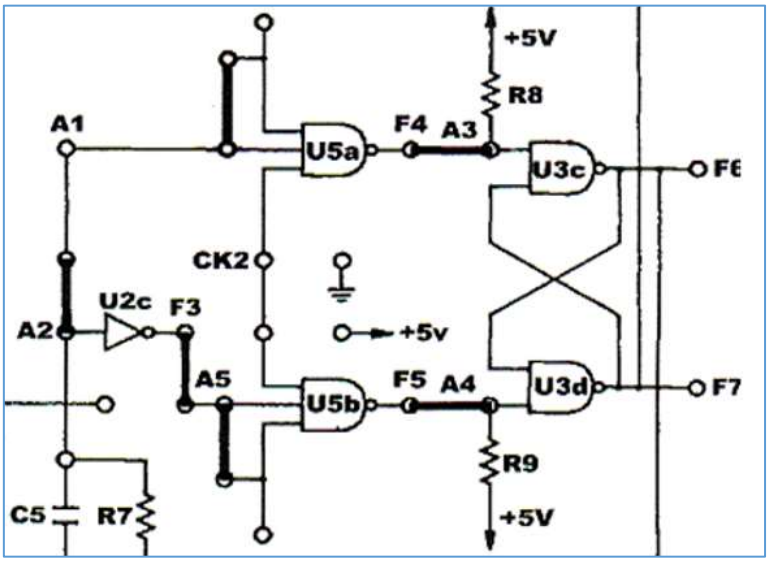



**Figure 4.5-b.** D Flip-Flop Implementation

### IV.4. Implementation of a JK Flip-Flop Based on RS

- Recall the truth table of the JK flip-flop, and then draw its logic diagram based on RS.
- Build the circuit on KL-33008 using Block d, and wire as in Figure 4.5-c.

- Connect the +5V (TTL) to the fixed supply, and ground both modules. Connect CK2 to the pulser switch output. Connect inputs J and K to the Data Switch outputs, and outputs F6 and F7 to logic indicators.
- Apply input sequences (J, K, and manual CK2 clock) and record the outputs in a table.

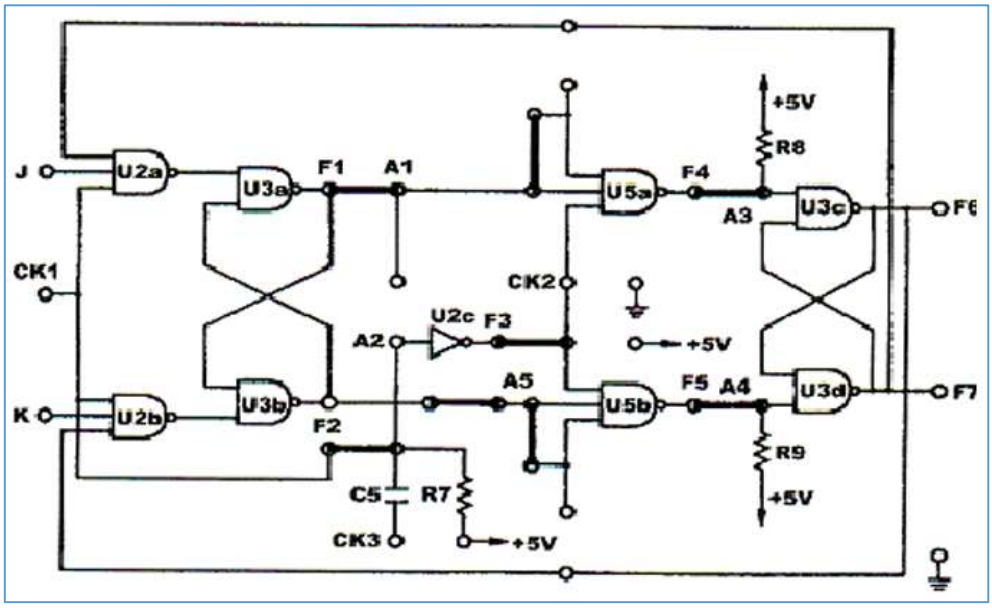



**Figure 4.5-c. JK Flip-Flop Implementation**

## V. Conclusion

Write a conclusion summarizing the objectives achieved, the main experimental steps completed, and the lessons learned during this lab.

In conclusion, the implementation of RS, D, and JK Flip-Flop circuits using the KL-33008 module has provided students with a practical understanding of sequential logic and data storage mechanisms in digital systems. Through experimentation and analysis of timing behavior, learners were able to observe how flip-flops respond to clock signals and control inputs to maintain stable output states. This hands-on experience reinforces theoretical concepts related to synchronous circuits and highlights the importance of flip-flops as fundamental building blocks in the design of registers, counters, and memory units. By completing this activity, students have strengthened their ability to interpret timing diagrams, configure sequential circuits, and evaluate their performance in real hardware environments.

# Workshop 5: Sequential Logic Circuits – Binary Counters and Down-counters

## I. Objectives

This activity introduces the concept of automatic counting sequences in digital systems through the design and implementation of binary counters and Down-counter s. Using the KL-33009 training module, students will explore how sequential logic circuits can be configured to increment or decrement binary values in response to clock pulses.

At the end of this lab, the student should be able to:

- Analyze the difference between synchronous and asynchronous counter architectures.
- Learn how to configure flip-flops to implement up-counters and down-counters using the KL-33009 training module.
- Understand the operation of binary counters and Down-counter s.
- Design and implement both synchronous and asynchronous circuits.

## II. Required Equipment

- KL-300 basic unit;
- Kl-33005 and KL-33009 modules;
- Digital clock pulse generator
- Set of logic probes or LEDs for output state indication
- Connecting wires and jumper cables
- External reset and enable switch inputs
- DC power supply compatible with the training module
- Timing diagram reference sheet or oscilloscope (optional for clock signal visualization)

## III. Theoretical Background

A counter or Down-counter is a sequential logic circuit composed of a set of n flip-flops interconnected through logic gates. It is used to count events or pulses, operating either with a synchronous or an asynchronous clock.

### III.1. Asynchronous Counter/Down-counter

An asynchronous counter, also called a ripple counter, is a system in which the clock signal

of each stage is provided by the output of the preceding stage.

Figure 5.1 illustrates an asynchronous counter implemented using JK flip-flops.

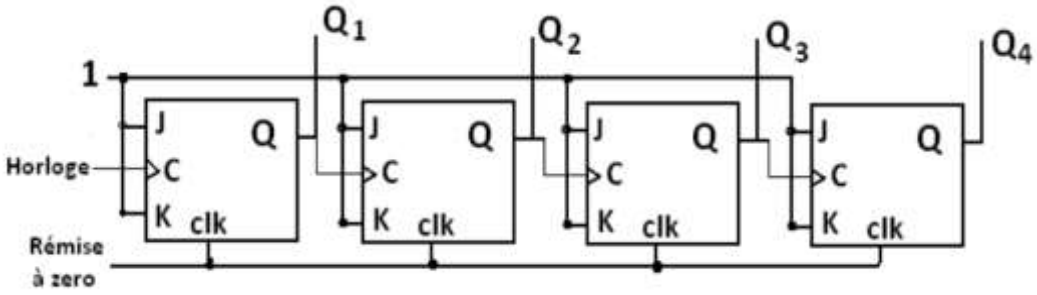



**Figure 5.1.** Asynchronous modulo-16 counter/Down-counter based on JK flip-flops

### III.2. Synchronous Counter/Down-counter

A synchronous counter applies the clock signal simultaneously to all flip-flop inputs. This eliminates issues related to propagation delays across stages, since all flip-flops change state at the same time.

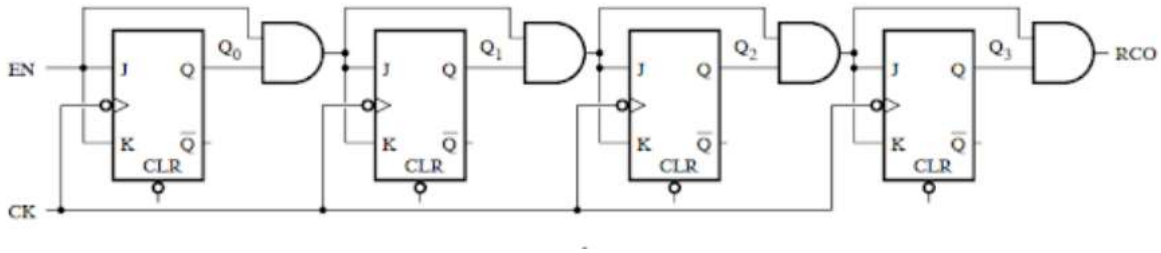



**Figure 5.2.** Synchronous modulo-16 counter/Down-counter based on JK flip-flops

### IV. Work requested

In this workshop, using the KL-33009 (see Figure 5.3) training module, students are required to implement an asynchronous binary up-counter based on JK flip-flops and verify the correct counting sequence from 0000 to 1111 (modulo-16). The circuit must then be reconfigured to operate as a binary down-counter to observe the reverse progression from 1111 back to 0000. Learners will analyze the propagation delay between flip-flops on the KL-33009 module to understand its impact on asynchronous counter behavior and integrate a reset control to force the system back to its initial state during operation. A modulo-10 (BCD) counter must also be designed by adding the necessary logic to reset the sequence after $1001_2$, followed by verification of the complete counting cycle. Output transitions should be observed through the module's LEDs or connected logic probes, and students must interpret the corresponding timing diagrams, compare the behavior of asynchronous and synchronous counters, and

troubleshoot any anomalies detected in the counting sequence while identifying and explaining their causes.

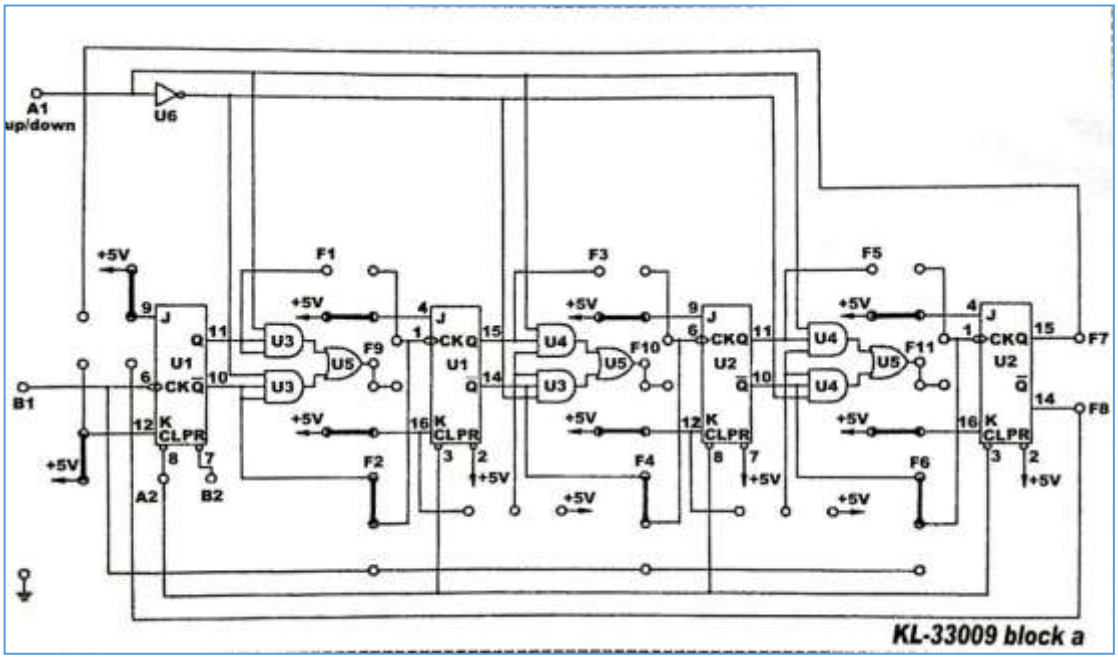



**Figure 5.3:** Implementation of a 4-bit Counter/Down-counter using KL-33009 module, Block a.

### IV.1. Asynchronous 4-Bit Binary Counter

Using the KL-33009 module, Block a, build the circuit shown in the figure 5.4-a.

- Connect A2 (Clear) to SW0, A1 (Up/Down) to +5V, B2 to +5V, and B1 (CK) to a clock signal with a frequency of 1 Hz.
- The outputs are F1, F3, F5, and F7.
- Initially, set SW0 = "0" to reset (initialize) the outputs, then switch SW0 = "1" to start the counting process.
- Finally, determine the modulus of this counter.

The circuit is based on JK flip-flops configured with **J = 1** and **K = 1** for each stage.

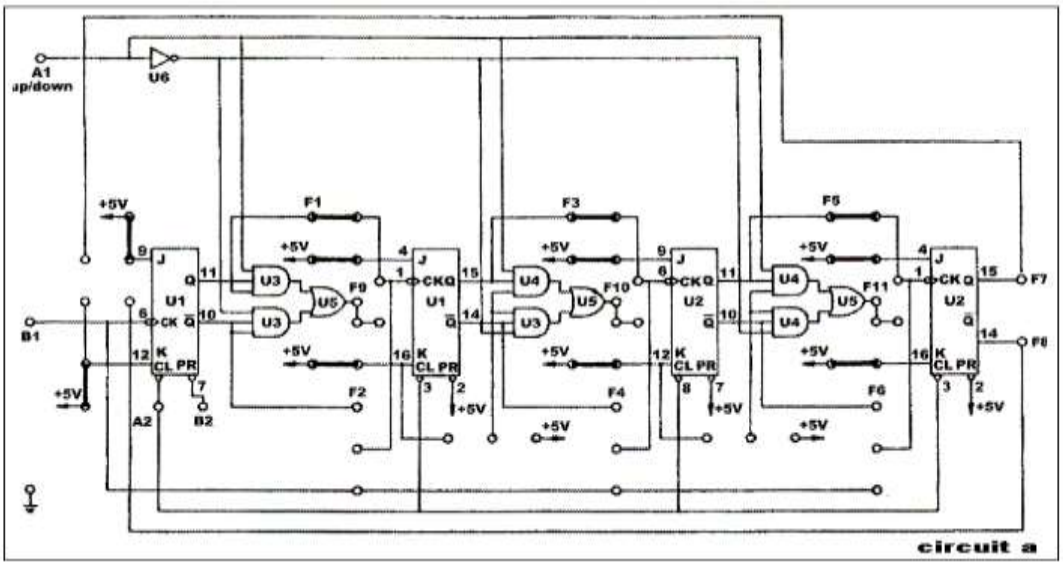



**Figure 5.4-a:** Implementation of an asynchronous counter using the KL-33009 module, Block a

### IV.2. Synchronous 4-Bit Binary Down-counter

- Use the KL-33009 module to build the required circuit, complete the connections on Block a, then insert the jumpers and wires as shown in Figure 5.1.
- Connect the +5V (TTL) terminal of the module to the +5V fixed power supply and connect the grounds. Then connect inputs A2 and A1 to the TTL outputs of the Data Switch, and outputs F1, F3, F5, and F7 to the logic indicators.
- Connect B1 (CK) to the TTL output of the pulser switch.
- Apply input sequences and record the output states in a table.

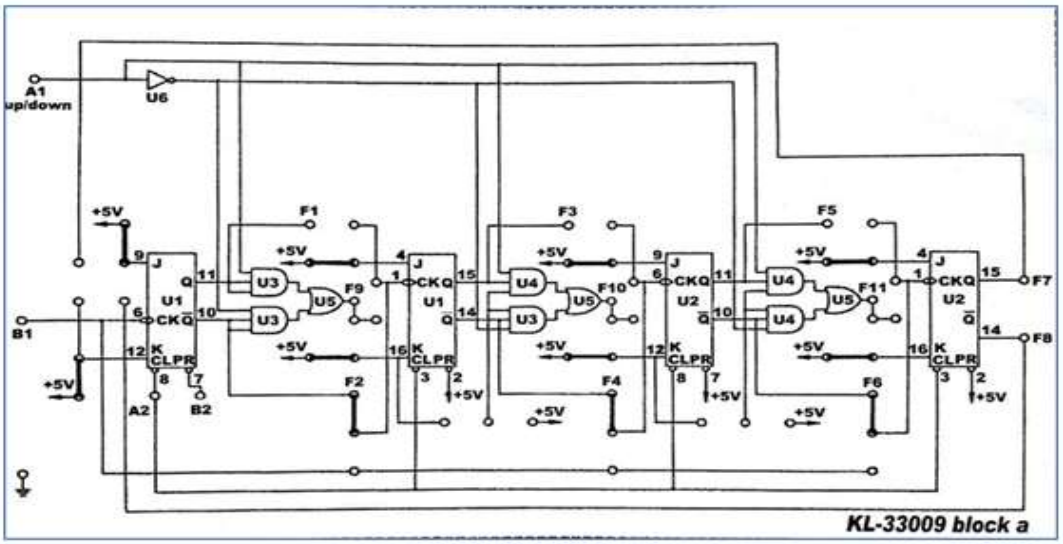



**Figure 5.4-b:** Implementation of a 4-bit binary Down-counter using Block a

### IV.3. Synchronous 4-Bit Binary Counter and Down-counter

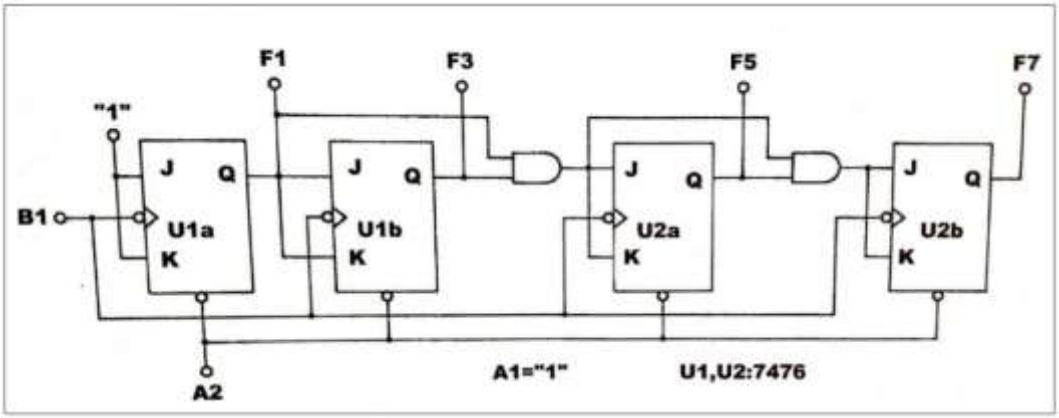



**Figure 5.5. Circuit of a 4-bit binary counter and** Down-counter

- Use the KL-33009 module (Block a) to build the required circuit, complete the connections as shown in Figure 5.5.
- Connect the +5V (TTL) terminal of the module to the +5V fixed power supply and connect the grounds. Then connect input A2 to the TTL outputs of the Data Switch, and outputs F1, F3, F5, and F7 to the logic indicators.
- Connect B1 (CK) to the Clock Generator output and set the frequency to 1 kHz.
- Set A1 to "1" and observe the outputs:
  - Record the output states in a table.
  - Capture the output signals using an oscilloscope on the timing diagram provided.
- Set A1 to "0" and observe the outputs:
  - Record the output states in a table.
  - Capture the output signals using an oscilloscope on the timing diagram provided.

## V. Conclusion

Write a conclusion summarizing the objectives achieved, the main experimental steps completed, and the lessons learned during this lab.

The implementation of binary counters and decounters using the KL-33009 module allowed students to practically explore the principles of sequential logic and state progression in digital systems. By working with JK flip-flop configurations, learners observed how counting sequences are generated and controlled through clock pulses, as well as how reset and enable signals influence system behavior. The comparison between up-counting and down-counting operations enhanced their understanding of reversible state transitions, while the design of a modulo-16 counter introduced the concept of custom sequence limitation through additional logic.

# CONCLUSION

Throughout this practical work handout, students were progressively introduced to the fundamental and advanced concepts of digital electronics through a structured series of workshops. Beginning with basic logic gates and Boolean algebra, the exercises gradually evolved toward more complex combinational and sequential circuits, including arithmetic units, data routing systems, flip-flops, and binary counters. Each activity was designed not only to reinforce theoretical knowledge but also to develop the students' ability to interpret logic diagrams, implement digital circuits using dedicated training modules, and analyze system behavior through experimental validation.

Through the use of the LAB KL-300 training system, students were able to explore in depth the fundamental concepts of digital electronics, from basic logic gates to more complex combinational and sequential circuits such as adders, multiplexers, flip-flops, and counters. Each experiment was carefully designed to reinforce theoretical knowledge acquired during lectures by applying it directly to real hardware implementations. This practical approach allowed students to bridge the gap between abstract Boolean logic and its physical realization in electronic systems.

By engaging in hands-on implementation and troubleshooting, students strengthened their understanding of digital logic design and gained essential technical skills useful for future engineering applications in automation, embedded systems, and control technologies. This practical experience serves as a solid foundation for more advanced studies in digital system design, microcontroller programming, and hardware-based automation. Ultimately, the successful completion of this set of workshops reflects the students' ability to connect theory with practice and prepares them for real engineering challenges where precision, logic, and systematic problem-solving are required.

## Bibliography

[1] Floyd, T. L. (2015). *Digital fundamentals* (11th ed.). Pearson Prentice Hall.

[2] Roth, C. H., & Kinney, L. L. (2013). *Fundamentals of logic design* (7th ed.). Cengage Learning.

[3] Tocci, R. J. (2010). *Digital systems: Principles and applications* (11th ed.). Pearson.

[4] Perrin, J. (2009). *Logique combinatoire et séquentielle: De la porte logique au système programmable*. Éditions Ellipses

[5] Dixit, J. B., & Dixit, A. (2010). *Digital systems: Design and applications*. Laxmi Publications.

[6] Texas Instruments. (2000–present). *7400 series logic family datasheets*. Texas Instruments Corporation.

[7] K&H MFG Co. Ltd. (2018). *KL-300/KL-330xx digital training module manuals*. K&H Educational Equipment Manufacturer.

[8] Tondo, C. L., & Gimpel, S. E. (2000). *Exercices corrigés sur le langage C* (2e éd.). Eyrolles.