Lab 4: 4-bit Synchronous UP/DOWN Counter

18th February 2025

Assignment 4: Hardware Lab CS224

Team Members:

Kavya Kumar Agarwal	Roll No: 230101053
Parth Sunil Aher	Roll No: 230101072
Dhruv Pansuriya	Roll No: 230101071
Durgesh Shelke	Roll No: 230101053

Introduction

Counters are fundamental components in digital systems, used to increment or decrement a count value in response to clock signals. They can be categorized primarily into two types: **Asynchronous Counters** and **Synchronous Counters**.

Asynchronous Counters

In an **Asynchronous Counter**, only the first flip-flop is driven by the main clock. The clock input of the subsequent flip-flops is triggered by the output of the previous flip-flop, creating a ripple effect. This leads to propagation delay through the flip-flops, making asynchronous counters slower. Because the flip-flops do not change states simultaneously, asynchronous counters operate at lower frequencies and are typically used for slower counting applications.

Synchronous Counters

In contrast, **Synchronous Counters** use a single global clock to drive all flip-flops simultaneously. This ensures that each flip-flop receives the clock signal at the same time, leading to parallel state changes. The primary advantage of synchronous counters is their ability to operate at higher frequencies due to the lack of propagation delay between flip-flops. As each flip-flop changes states synchronously with the clock, they are also referred to as *parallel counters*. Their ability to operate faster and more precisely makes them ideal for high-speed digital applications.

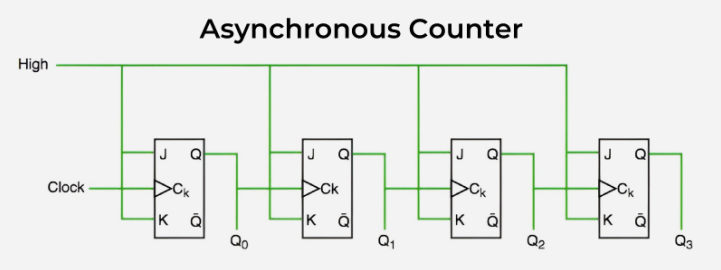

Figure 1: Asynchronous-Counter

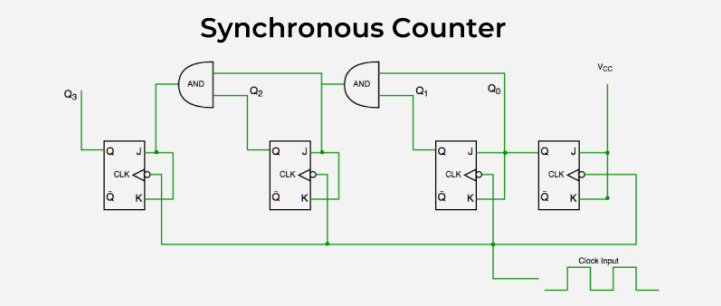

Figure 2: Synchronous-Counter

Our Project Description: 4-bit Synchronous UP/DOWN Counter

Project Overview

In this project, we are tasked with designing and implementing a **4-bit Synchronous Binary UP/DOWN Counter** using a breadboard and integrated circuits (ICs). This counter will serve as a demonstration of a synchronous counter and will showcase its capability to count both upwards and downwards.

Functional Description

The 4-bit synchronous counter will have the following inputs and features:

- **Clock Input:** A clock signal that triggers the counter to increment or decrement.
- **Reset Input (1-bit):** A reset signal that resets the counter to 0. The reset has the highest priority over all other inputs.
- **Set Input (1-bit):** A set signal that allows setting the counter to a specified 4-bit value. The set signal has higher priority over the *up/down* control.
- **Up/Down Input (1-bit):** A control signal that determines whether the counter counts up (1) or down (0).

The counter will count from **0 to 15**, then reset back to **0** in a continuous loop. Upon activation of the reset signal, the counter will reset to **0** at the next clock pulse. The set signal allows setting the counter to a specific 4-bit value. The *up/down* input controls whether the counter counts upwards or downwards. The reset signal has the highest priority, followed by the set signal, and finally, the up/down control.

For the demonstration, the breadboard's automatic clock will be used, and the counter's outputs will be displayed on LEDs.

Truth Table and Behavioral Analysis

Truth Table and Behavioral Analysis

We created a truth table with columns for the 4 bits Q_0, Q_1, Q_2, Q_3 and their next states Q_0', Q_1', Q_2', Q_3', as well as the JK inputs $(J_0, K_0, J_1, K_1, J_2, K_2, J_3, K_3)$ of each flip-flop. The control signal *up* (which determines whether the counter counts up or down) was also considered in the analysis.

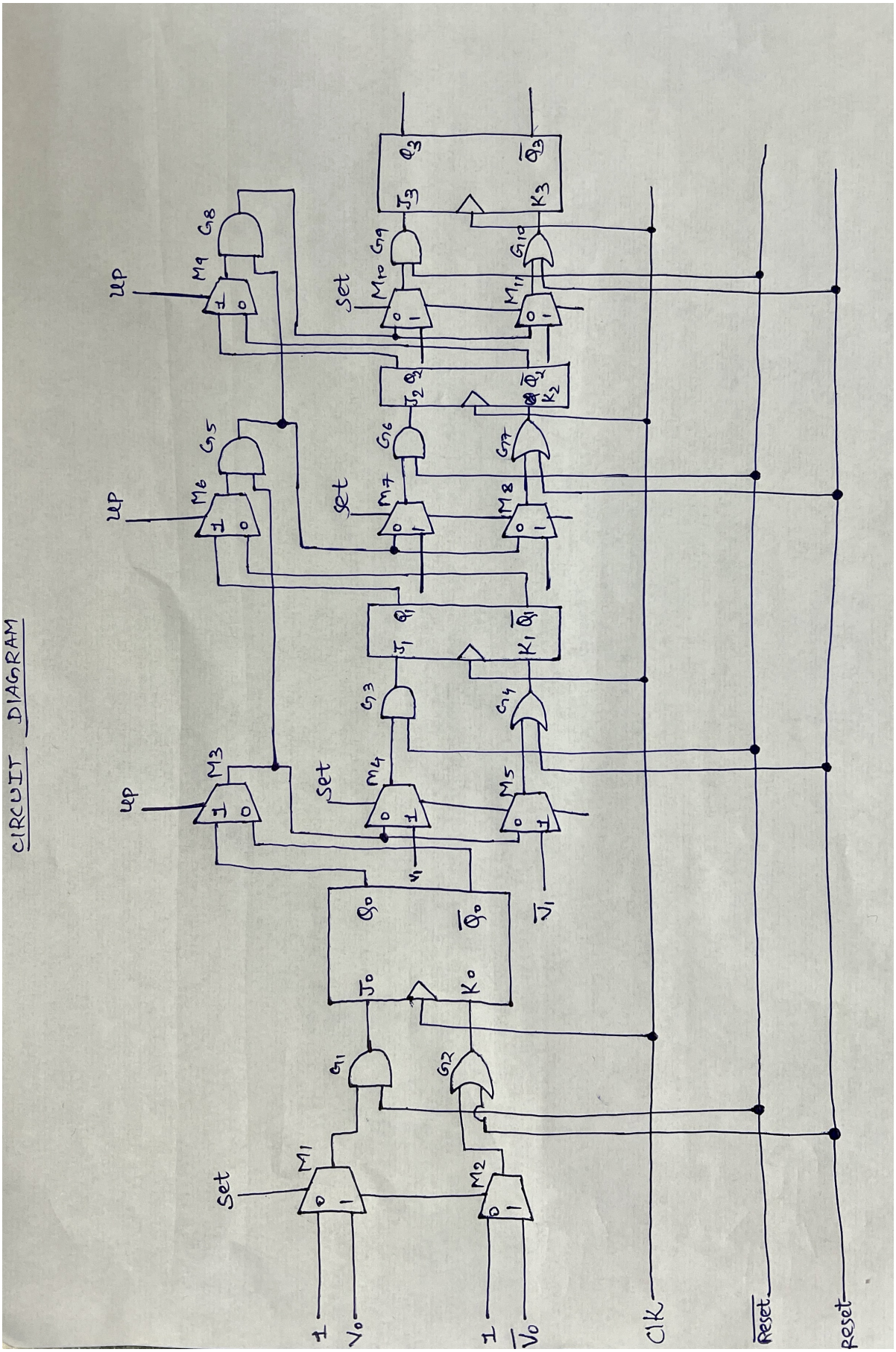

Figure 3: Circuit Diagram

Truth Table 1: Synchronous Counter UP Operation

Up	Q_3	Q_2	Q_1	Q_0	Q'_3	Q'_2	Q'_1	Q'_0	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
1	0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
1	0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
1	0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
1	0	1	0	0	0	1	0	1	0	X	0	X	0	X	1	X
1	0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
1	0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
1	0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	1	0	0	1	1	0	1	0	X	0	0	X	1	X	X	1
1	1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	X
1	1	0	1	1	1	1	0	0	X	0	1	X	X	1	X	1
1	1	1	0	0	1	1	0	1	X	0	X	0	0	X	1	X
1	1	1	0	1	1	1	1	0	X	0	X	0	1	X	X	1
1	1	1	1	0	1	1	1	1	X	0	X	0	X	0	1	X
1	1	1	1	1	0	0	0	0	X	0	X	0	X	0	X	0

Table 1: Truth Table 1: Synchronous Counter UP Operation.

Truth Table 2: Synchronous Counter DOWN Operation

Up	Q_3	Q_2	Q_1	Q_0	Q'_3	Q'_2	Q'_1	Q'_0	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
0	1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
0	1	0	0	1	1	0	1	0	X	0	0	X	1	X	X	1
0	1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	X
0	1	0	1	1	1	1	0	0	X	0	1	X	X	1	X	1
0	1	1	0	0	1	1	0	1	X	0	X	0	0	X	1	X
0	1	1	0	1	1	1	1	0	X	0	X	0	1	X	X	1
0	1	1	1	0	1	1	1	1	X	0	X	0	X	0	1	X
0	1	1	1	1	0	0	0	0	X	1	X	1	X	1	X	1

Table 2: Truth Table 2: Synchronous Counter DOWN Operation.

article amsmath

State Transition Logic

We derived the logic expressions for the JK inputs of each flip-flop to implement the set and reset functionalities. The logic equations for each flip-flop's JK inputs are based on the current state, control signal "up", and set/reset inputs.

Basic Logical Expressions

The following table shows the logical expressions for the JK inputs of the flip-flops:

Flip-Flop	**Logical Expression**
J_0	1
J_1	$Q_0(\text{up}) + Q_0'(\text{up}')$
J_2	$Q_1Q_0(\text{up}) + Q_1'Q_0'(\text{up}')$
J_3	$Q_2Q_1Q_0(\text{up}) + Q_2'Q_1'Q_0'(\text{up}')$
K_0	1
K_1	$Q_0(\text{up}) + Q_0'(\text{up}')$
K_2	$Q_1Q_0(\text{up}) + Q_1'Q_0'(\text{up}')$
K_3	$Q_2Q_1Q_0(\text{up}) + Q_2'Q_1'Q_0'(\text{up}')$

These logical expressions define the inputs for the JK flip-flops, allowing the 4-bit synchronous UP/DOWN counter to operate correctly under the given conditions. The equations incorporate the control signal 'up' to determine whether the counter will increment or decrement at each clock cycle, as well as the reset and set conditions.

Set and Reset Behavior Using MUX

In digital circuits, particularly in the context of up-counters, the proper management of flip-flops (FFs) using the *Set* and *Reset* signals is crucial for ensuring the counter operates as intended. These signals serve as control inputs to the flip-flops, dictating whether they should retain their current state, be reset to a known value, or be set to a specific value. The behavior of these control signals is typically governed by multiplexers (MUX), which allow for flexible routing of different inputs based on the current state of the control signals. By analyzing how the flip-flops behave under different *Set* and *Reset* conditions, we can better understand how to manipulate the counter's state.

In this section, we will explore the interaction of *Set* and *Reset* signals and their impact on the flip-flops' operation. We will examine four distinct cases, each corresponding to different combinations of *Set* and *Reset* values. The analysis will focus on how these combinations affect the state of the counter and the behavior of the flip-flops. Understanding this behavior is essential for designing reliable up-counters and ensuring that they respond correctly to control signals in various scenarios.

Case 1: `Set` = 0 and `Reset` = 0

In this first case, both the *Set* and *Reset* signals are 0. This is the simplest scenario where neither signal is active. The flip-flops remain in their current state and no changes occur unless there is a future transition of the control signals. Specifically:

- For both FF1 and FF2, the inputs `J` and `K` will receive the value of `Q0`, which is the current state of the flip-flop.
- Since both `Set` and `Reset` are 0, the flip-flops are effectively "locked" in their current state. This means that the flip-flops will not change their values until a new signal transition occurs.

This case represents a holding or "no-action" state, where the system is stable and the counter remains unchanged, ensuring that no unintended transitions occur.

Case 2: `Set` = 0 and `Reset` = 1

In this second case, the *Set* signal remains 0, but the *Reset* signal is set to 1. This combination has a significant effect on the behavior of the flip-flops, as the reset condition is active. Specifically:

- For both FF1 and FF2, the input `J` will be 0, and `K` will be 1. This combination of `J` and `K` forces the flip-flops to reset.
- When a reset operation is triggered, the output of the flip-flop is set to 0, regardless of its previous state. This means that both flip-flops will clear, and the counter will return to the initial state of 0000.

The reset operation is particularly useful in cases where the system needs to start over, such as after a certain event or condition has been met. It is a vital feature in any counter design, providing a reliable way to clear the previous value and prepare the system for a fresh cycle.

Case 3: `Set` = 1 and `Reset` = 0

In this case, the *Set* signal is 1, while the *Reset* signal is 0. This combination allows the flip-flops to be set to a specific value. Specifically:

- For both FF1 and FF2, the input `J` will receive `V1`, and `K` will receive the complement of `V1`, denoted as `V1_bar`.
- The flip-flop will be set to a state determined by the value of `V1`. If `V1` is 1, the flip-flop will be set to 1. Conversely, if `V1` is 0, the flip-flop will be set to 0.
- This condition allows for precise control over the state of the flip-flops, enabling the counter to take on specific values depending on the input.

The *Set* operation plays a key role when specific values need to be loaded into the counter. It ensures that the counter can be initialized or re-initialized to a known state based on the value of `V1`.

Case 4: `Set` = 1 and `Reset` = 1

The final case involves both the *Set* and *Reset* signals being active, with both signals set to 1. This situation could potentially lead to conflicting actions, as the *Set* signal typically asserts a value while the *Reset* signal forces the output to 0. However, in most flip-flop designs, the reset signal has priority over the set signal. Specifically:

- For both FF1 and FF2, `J` will be 0, and `K` will be 1, which forces the flip-flops to reset, regardless of the active *Set* signal.
- The priority of the *Reset* signal ensures that the counter will be cleared, returning the flip-flops to the 0 state.

In systems where both signals might be triggered at the same time, ensuring that the reset signal has higher priority provides a safety mechanism, guaranteeing that the system does not enter an undesirable state.

Conclusion

In summary, the behavior of the *Set* and *Reset* signals in an up-counter using multiplexers plays a crucial role in managing the state of the flip-flops and ensuring the proper operation of the counter. Through the analysis of these four cases, we can see how the control signals influence the state of the flip-flops and how different combinations of *Set* and *Reset* can be used to achieve various counter behaviors.

article graphicx

Number of Gates Used and IC Names

The following components are used in the design of the circuit, along with the corresponding ICs and the number of ICs required:

AND Gates

- **Number of AND gates:** 6
- **IC Used:** 7408 (Quad 2-input AND gate)
- **Number of ICs used:** 2

OR Gates

- **Number of OR gates:** 4
- **IC Used:** 7432 (Quad 2-input OR gate)
- **Number of ICs used:** 1

JK Flip-Flops

- **Number of JK Flip-Flops:** 4
- **IC Used:** 74109 (Dual JK Flip-Flop)
- **Number of ICs used:** 2

2-to-1 Multiplexers

- **Number of 2-to-1 MUX:** 4
- **IC Used:** 74157 (Quad 2-input multiplexer)
- **Number of ICs used:** 3

Component Summary Table

To summarize the number of components and ICs used in the design, the following table is provided:

Component	IC Used	Number of ICs Used	Number of Components
AND Gates	7408 (Quad 2-input AND gate)	2	6
OR Gates	7432 (Quad 2-input OR gate)	1	4
JK Flip-Flops	74109 (Dual JK Flip-Flop)	2	4
2-to-1 MUX	74157 (Quad 2-input multiplexer)	3	4

Table 3: Summary of Components, ICs, and Counts

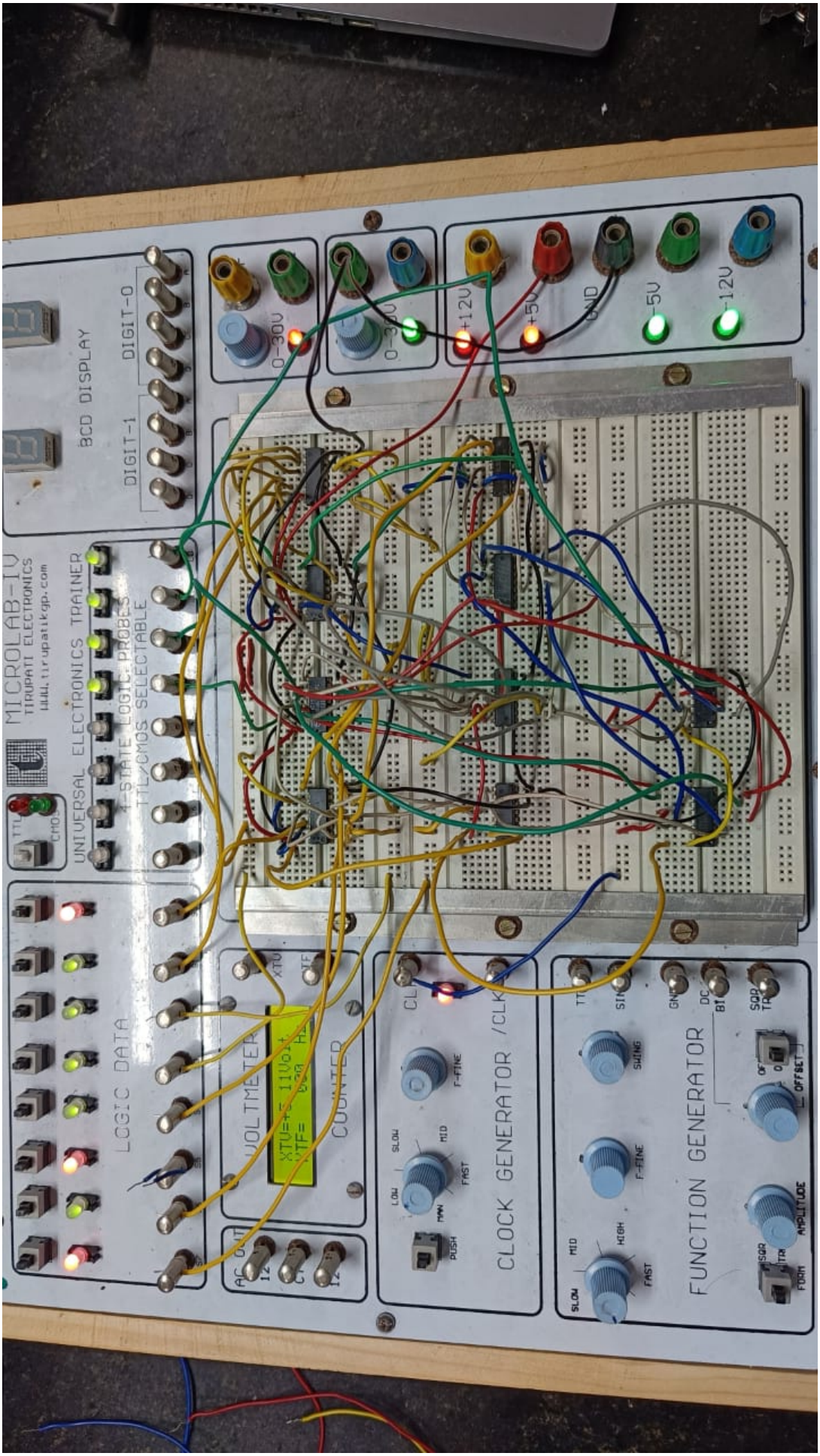

Figure 4: BreadBoard Image

1 ASM Implementation

The Algorithmic State Machine (ASM) approach provides a structured way to design sequential circuits, including an Up-Down Counter. It represents the system using ASM charts, which combine state diagrams, flowcharts, and truth tables.

1.1 ASM Chart for Up-Down Counter

The ASM chart consists of three main elements:

- **State Boxes:** Represent the current state of the counter.
- **Decision Boxes:** Define conditions based on control inputs (T, Set, Reset).
- **Conditional Outputs:** Specify actions such as incrementing or decrementing the counter.

1.2 State Definitions

- **Idle State:** The counter waits for the start signal.
- **Reset State:** If Reset = 1, the counter is cleared.
- **Set State:** If Set = 1, the counter is assigned a predefined value.
- **Counting State:** Based on the Up/Down control signal:
  - If Up = 1, count increments.
  - If Down = 1, count decrements.

1.3 ASM Table Representation

The ASM table lists transitions based on inputs:

1.4 Advantages of ASM Approach

- **Clearer Representation:** Combines state diagram logic with structured transitions.
- **Scalability:** Can be extended for multi-bit counters.
- **Ease of Hardware Implementation:** Can be directly converted into a hardware state machine.

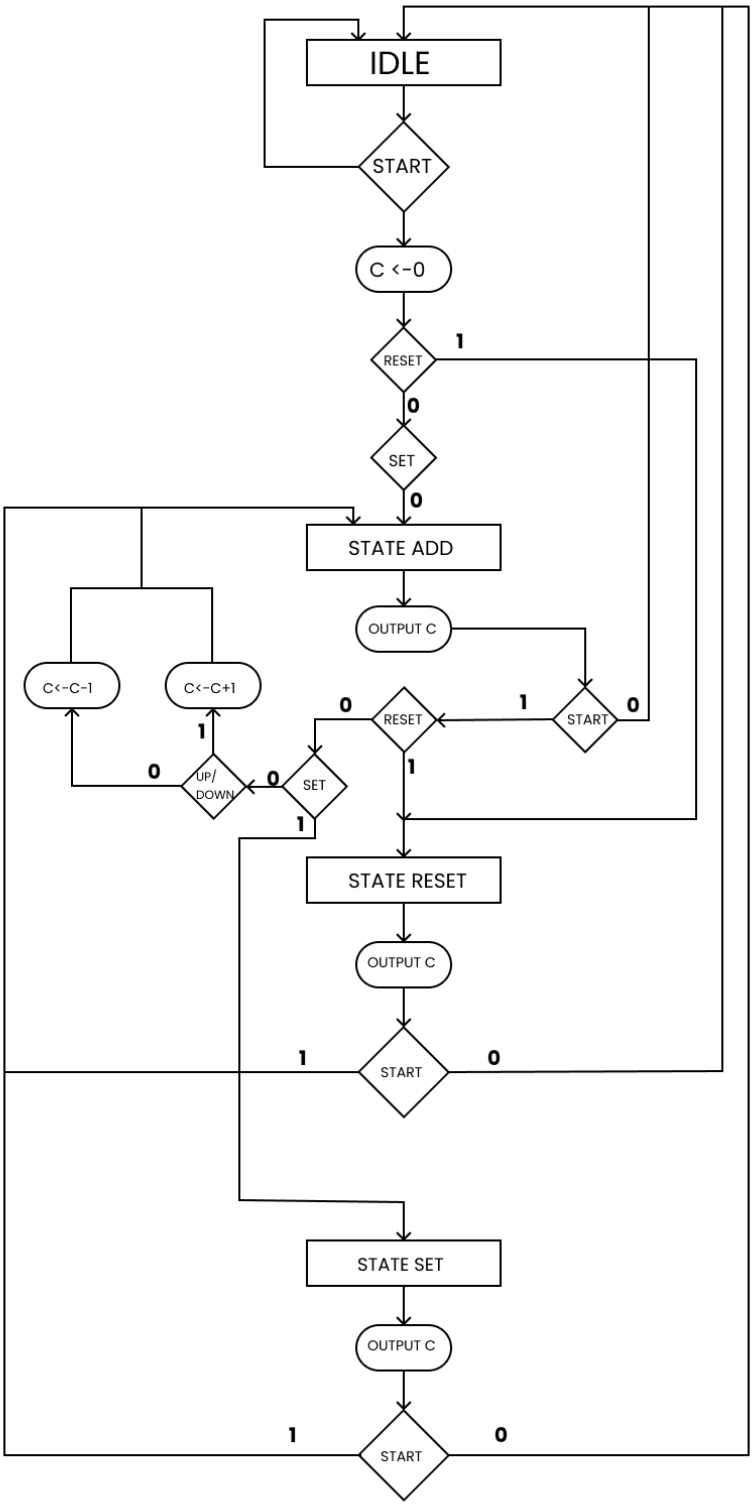

Figure 5: ASM CHART

2 ICs Used in the Project

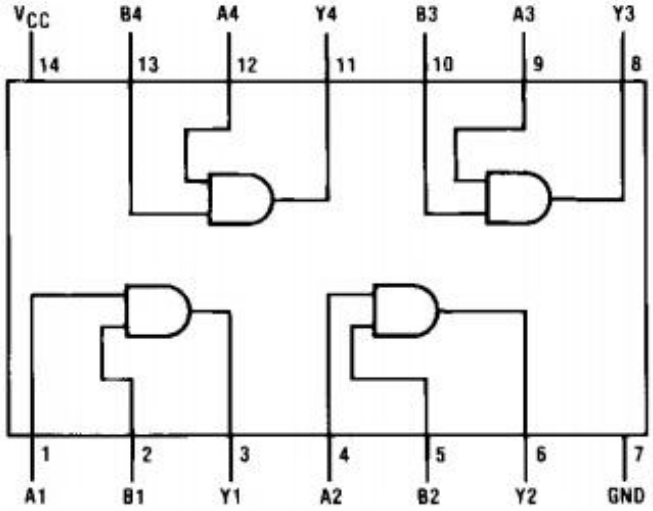

Figure 6: AND IC

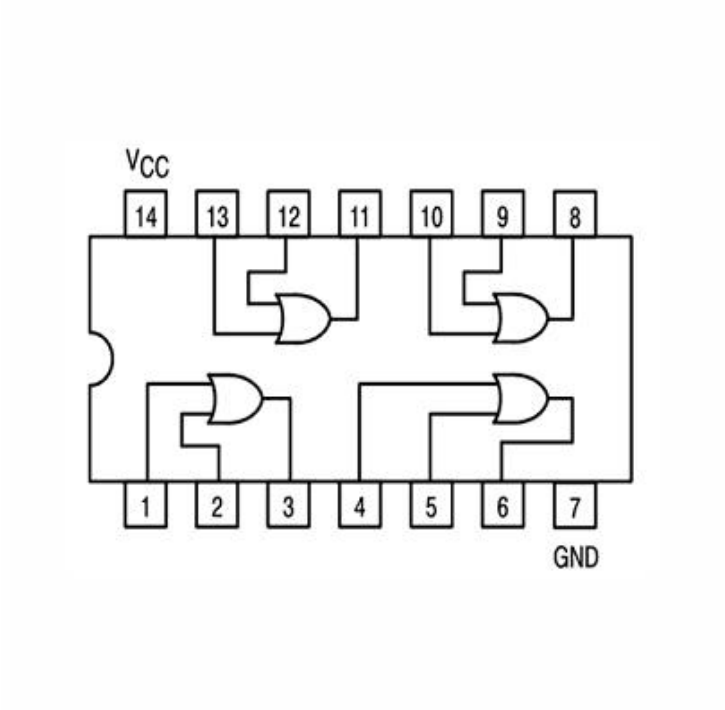

Figure 7: OR IC

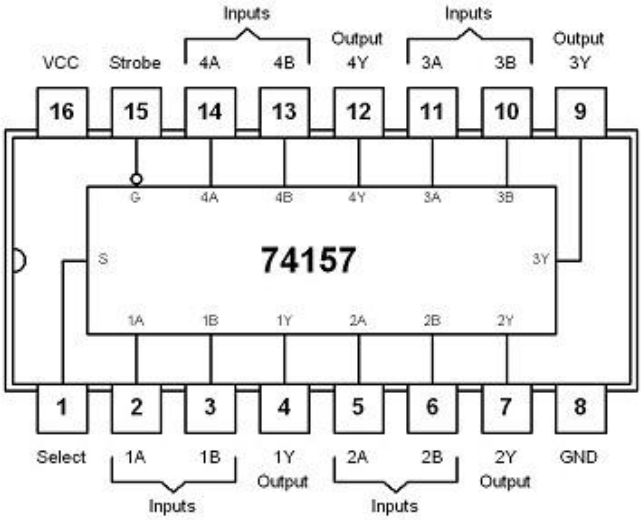

Figure 8: MUX IC

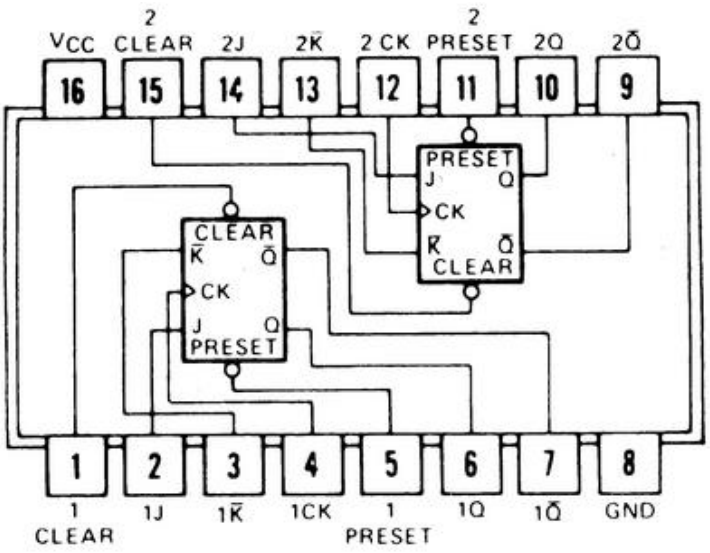

Figure 9: FF IC (Flip-Flop IC)

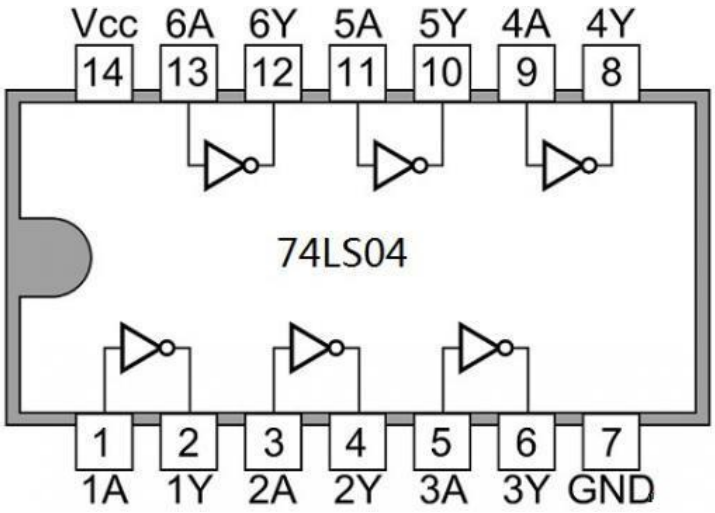

Figure 10: NOT IC