Design and Implementation of a Mod-7 Asynchronous Counter using T Flip-Flops

EE24BTECH11024 Abhimanyu Koushik Garimella
EE24BTECH11002 Agamjot Singh
IIT Hyderabad

March 27, 2025

1 Aim

To design and implement a Mod-7 Asynchronous Counter using T Flip-Flops (constructed from JK Flip-Flops) and to demonstrate its performance using a Cathode Ray Oscilloscope (CRO) with a clock signal generated from an Arduino, while also visualizing the counter states using LEDs.

2 Equipment Required

- JK Flip-Flops (IC 7476) - 3 units
- 3-input NAND Gate (IC 7410) - 1 unit
- Arduino board (for clock generation and power)
- Cathode Ray Oscilloscope (CRO)
- LED - 3 units (for visual representation of counter states)
- Resistors (560Ω) - 3 units (for LED current limiting)
- Breadboard and connecting wires

3 Theory

3.1 T Flip-Flop

A T (Toggle) Flip-Flop is a digital circuit that changes its output state (toggles) when triggered by a clock pulse if its T input is set to 1. If T=0, the flip-flop maintains its previous state. The characteristic equation of a T flip-flop is:

$$Q_{next} = Q \oplus T \tag{1}$$

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

3.2 JK Flip-Flop as T Flip-Flop

A JK Flip-Flop can be configured as a T Flip-Flop by connecting both J and K inputs to the same signal (T). When J=K=1, the flip-flop toggles with each clock pulse, which is the behavior of a T flip-flop with T=1.

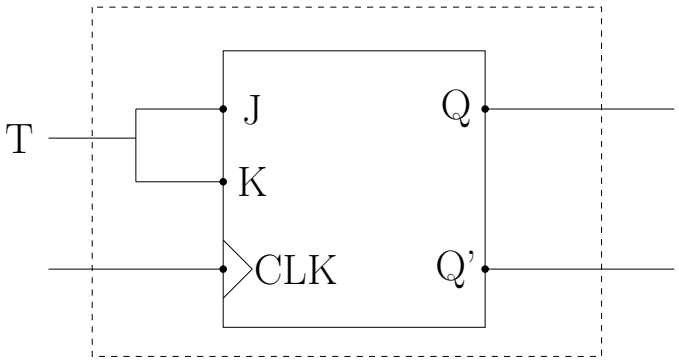

Figure 1: Constructing T Flipflop using JK Flipflop

3.3 Asynchronous Counter

In an asynchronous counter, the flip-flops do not change states simultaneously. The output of one flip-flop serves as the clock input for the next flip-flop. This creates a

ripple effect where the clock propagates through the counter, hence the name "ripple counter."

3.4 Mod-7 Counter

A Mod-7 counter cycles through 7 states (from 0 to 6) before resetting. For a binary counter, we need 3 flip-flops to represent states 0 to 6 (000 to 110 in binary). To ensure the counter resets after state 6 (110) and skips state 7 (111), we implement a reset mechanism using a NAND gate.
All the outputs (Q_1, Q_2, Q_3) are connected to a 3-input NAND gate and the output of the NAND gate is connected to the clear pin of the flipflop. When state 7 (111) is attained, the ouput of the NAND gate is 0 and state is switched to 0 (000) instead of 7 immediately.

4 Circuit and Timing Diagram

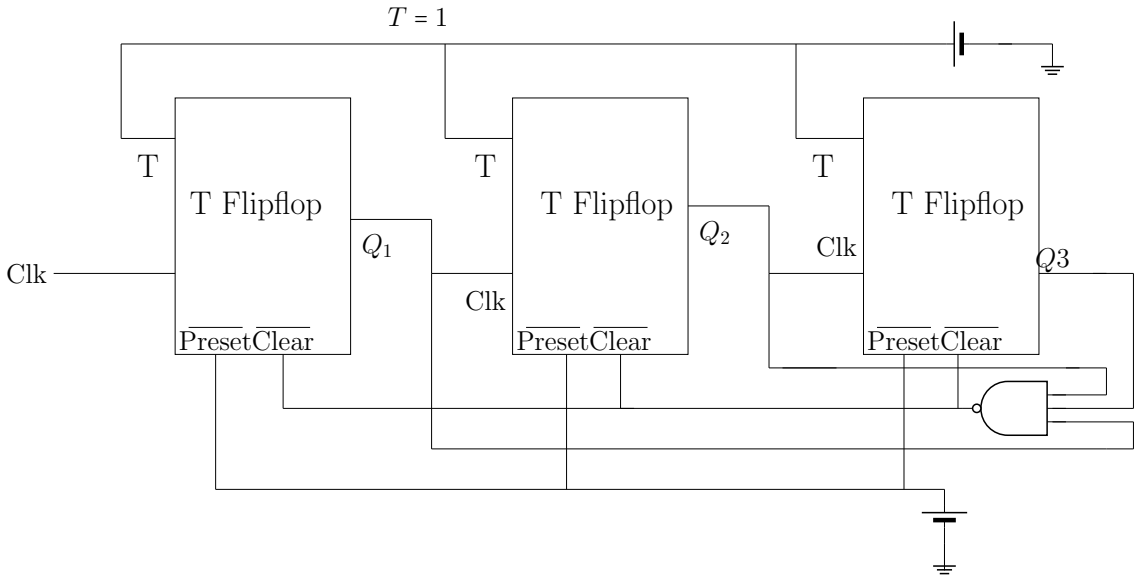

Figure 2: Implementation of a Mod-7 Asynchronous Ripple Counter

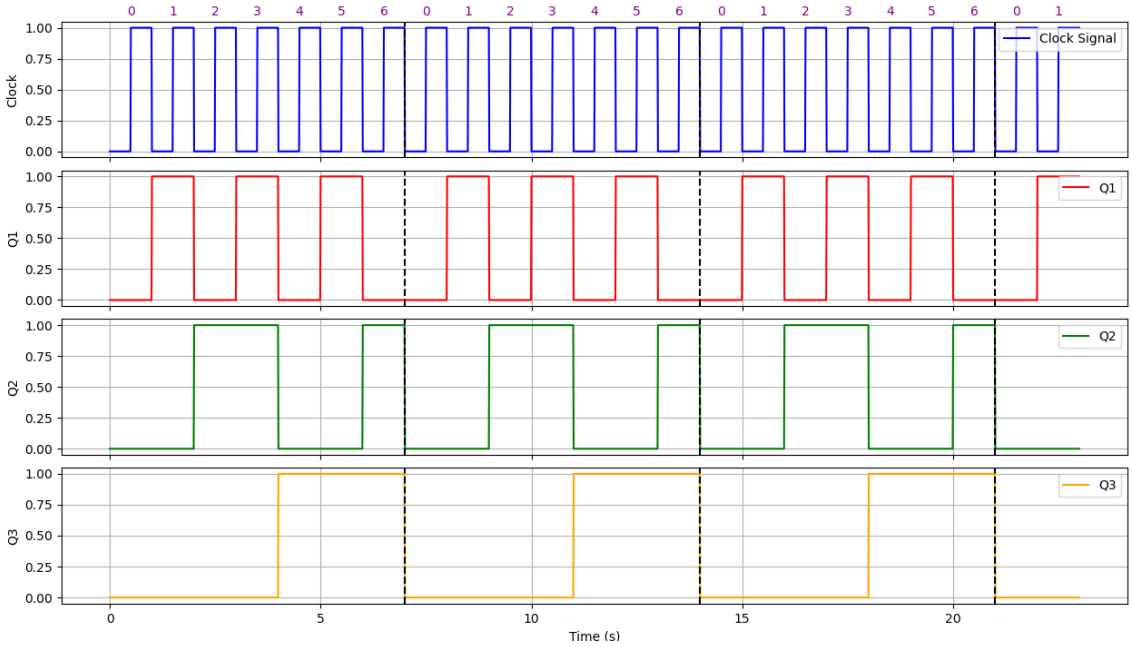

Figure 3: Timing Diagrams of Clock and all three outputs (Q_1, Q_2, Q_3)

5 Procedure

1. Configure the JK flip-flops as T flip-flops by connecting both J and K inputs to logic 1 (+5V).
2. Connect the flip-flops in cascade, with the Q output of each flip-flop connected to the clock input of the next flip-flop.
3. Connect the Arduino's pin 13 to the clock input of the first flip-flop (FF0).
4. Connect the Q outputs of all three flip-flops (Q0, Q1, Q2) to the inputs of the 3-input NAND gate (7410).
5. Connect the output of the NAND gate to the $\overline{\text{CLEAR}}$ pins of all flip-flops.
6. Connect the $\overline{\text{PRESET}}$ pins of all flip-flops to power (+5V).
7. Connect LEDs with current-limiting resistors to the Q outputs of each flip-flop for visual state representation.
8. Connect the Q outputs of all flip-flops to the CRO channels for visualization.

9. Program the Arduino to generate a square wave clock signal (code provided below).
10. Power up the circuit and observe the LED states and waveforms on the CRO.
11. Verify that the counter cycles through states 0 to 6 and then resets to 0.

5.1 Arduino Code for Clock Generation

```
void setup() {
  pinMode(13, OUTPUT);  // Set pin 13 as output
}

void loop() {
  digitalWrite(13, HIGH);  // Set pin 13 HIGH
  delay(500);              // Wait for 500 milliseconds
  digitalWrite(13, LOW);   // Set pin 13 LOW
  delay(500);              // Wait for 500 milliseconds
}
```

6 Results

6.1 LED States

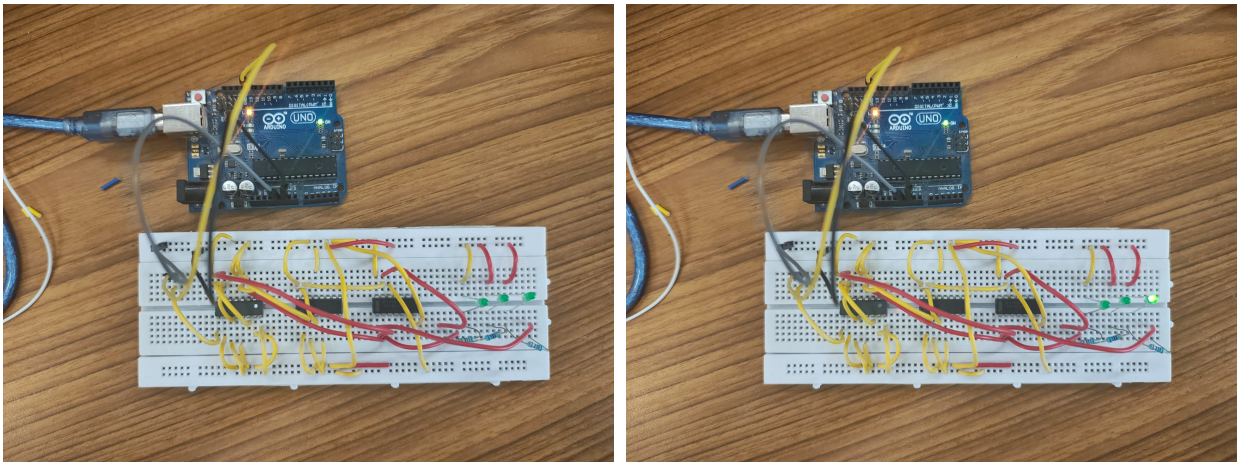

Figure 4: State 0 and 1

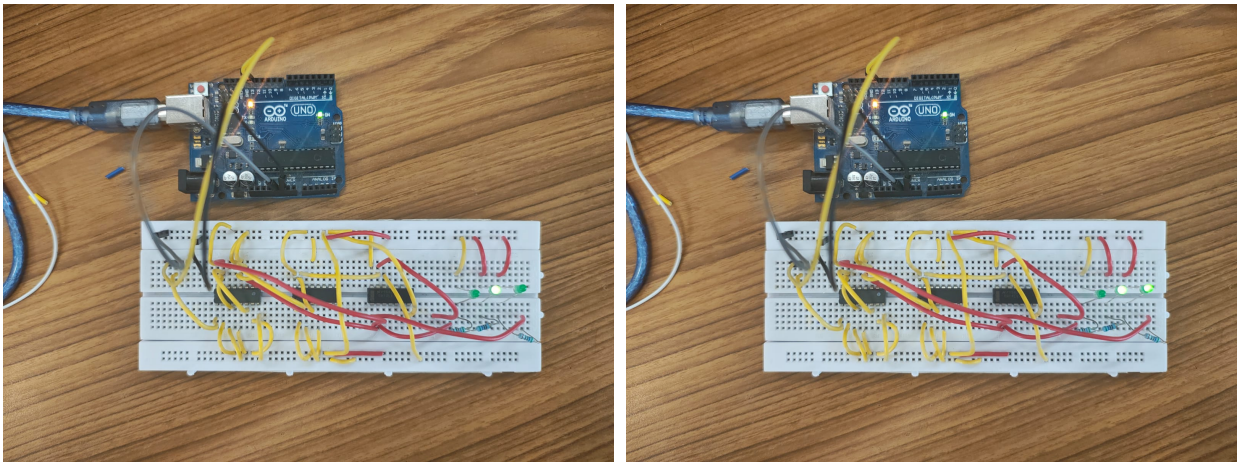

Figure 5: State 2 and 3

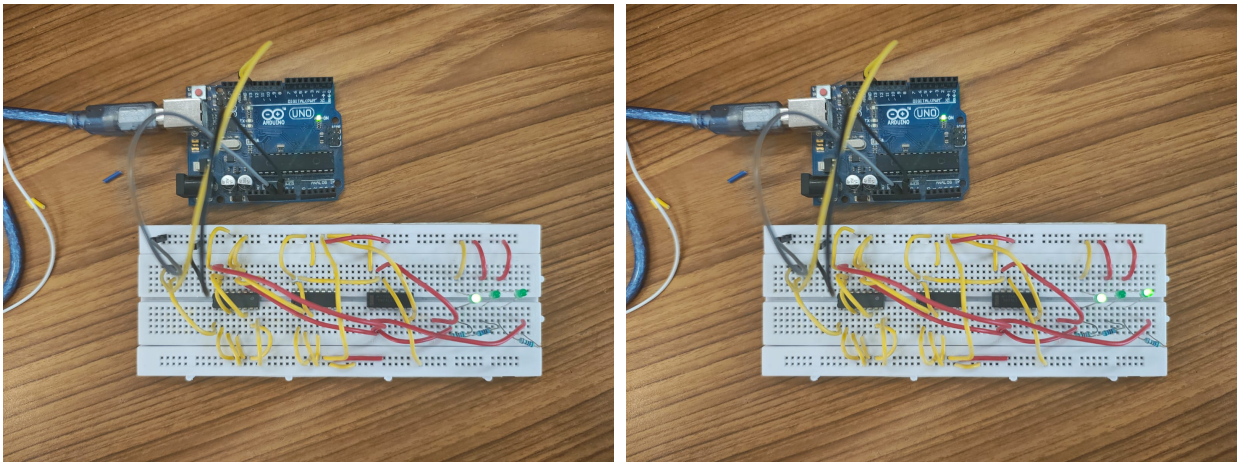

Figure 6: State 4 and 5

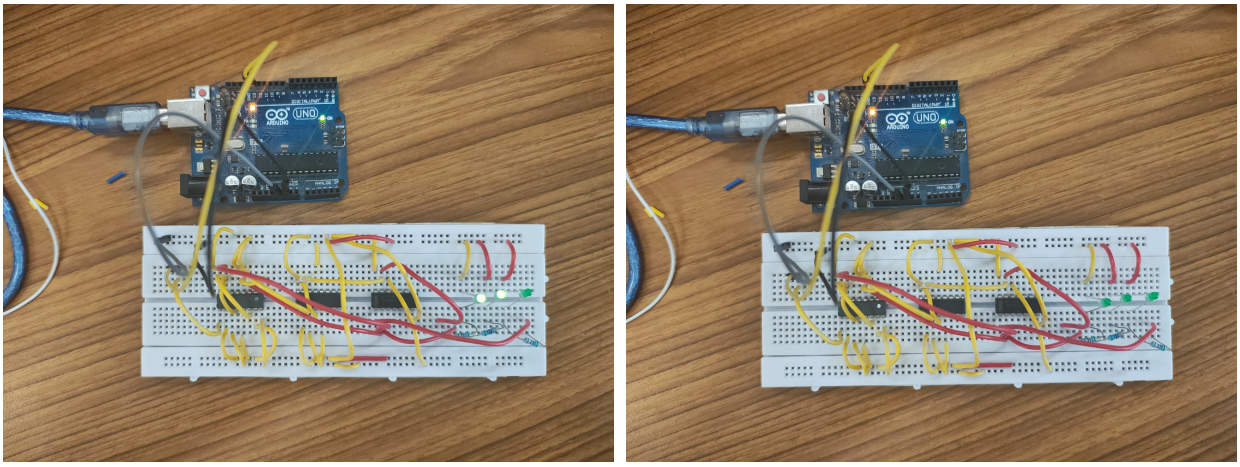

Figure 7: State 6 and back to 0

6.2 Oscilloscope Readings

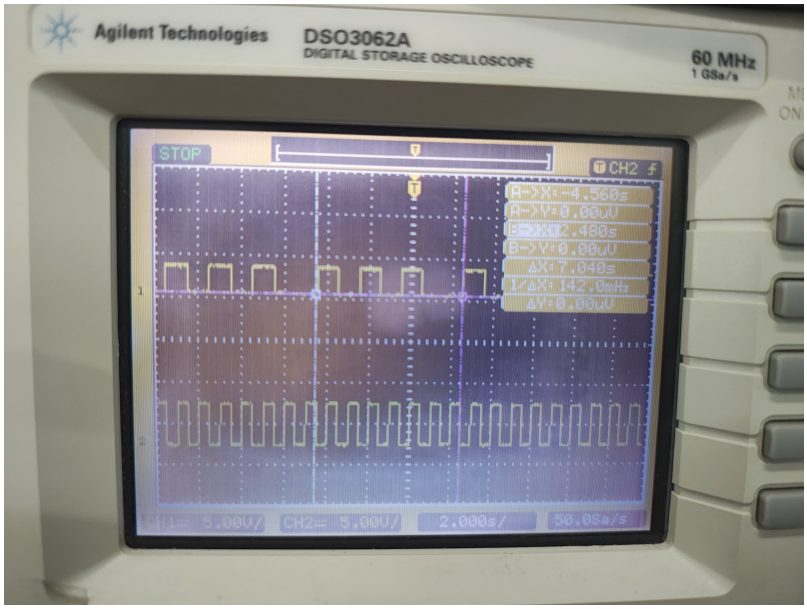

Figure 8: Reading of Q_1 vs Clock

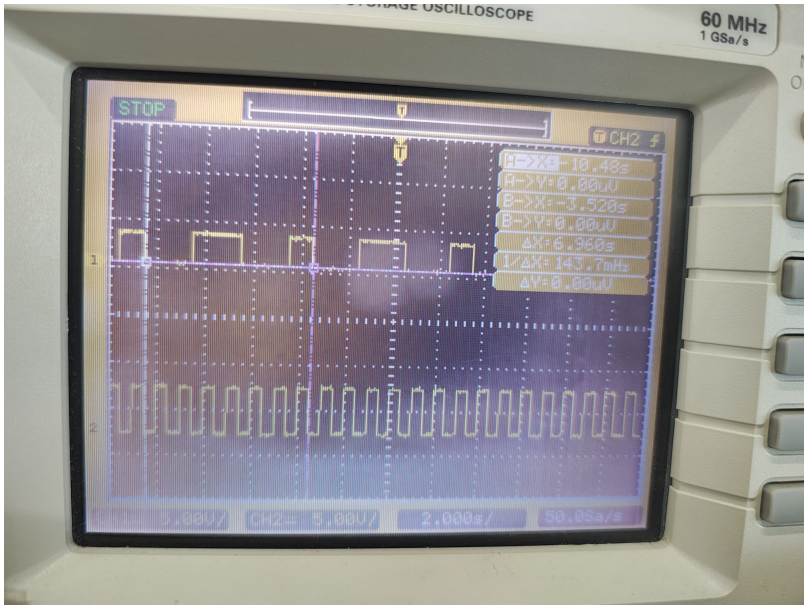

Figure 9: Reading of Q_2 vs Clock

Figure 10: Reading of Q_3 vs Clock

7 Observation

The counter was observed to cycle through the following states:

Count	Q2	Q1	Q0	Decimal Equivalent	LED State
0	0	0	0	0	Off-Off-Off
1	0	0	1	1	Off-Off-On
2	0	1	0	2	Off-On-Off
3	0	1	1	3	Off-On-On
4	1	0	0	4	On-Off-Off
5	1	0	1	5	On-Off-On
6	1	1	0	6	On-On-Off
7	0	0	0	0 (Reset)	Off-Off-Off

Table 1: State Sequence of Mod-7 Counter with LED Indications

On the CRO, the following waveforms were observed:

- Q_0: Square wave with the time period = 7s
- Q_1: Square wave also with the time period = 7s
- Q_2: Square wave also with the time period = 7s

The NAND gate output remained HIGH during states 0 through 6. When the counter attempted to enter state 7 (111), the NAND gate output went LOW momentarily, clearing all flip-flops and resetting the counter to state 0 (000).

The LEDs provided a visual representation of the counter states, with each LED corresponding to a bit in the counter (Q0, Q1, Q2). This allowed for easy verification of the counter's operation without the need for measurement equipment.

8 Conclusion

The Mod-7 asynchronous counter using T flip-flops was successfully designed and implemented. The counter correctly cycled through states 0 to 6 before resetting to state 0, skipping state 7 as intended. The timing diagram observed on the CRO matched the expected behavior, confirming the proper operation of the counter.

For applications requiring precise timing, synchronous counters would be more appropriate to eliminate the propagation delay issues inherent in asynchronous designs. However, the simplicity and ease of implementation of asynchronous counters make them suitable for many practical applications where slight timing discrepancies are acceptable.