MOD-7 Asynchronous Counter using T Flip-Flops

Design, Implementation and Testing Report

Krishna Patil - EE24BTECH11036
Deepak Ahirwar - EE24BTECH11014

March 27, 2025

Abstract

Abstract

The MOD-7 asynchronous counter successfully counts from 0 to 6 before automatically resetting to 0. This report details the design, implementation, and testing of the counter using JK flip-flops configured as T flip-flops, with an Arduino providing the clock signal. The counter displays its state on a 7-segment display and its operation is verified through oscilloscope measurements.

Contents

1 Theoretical Background

1.1 T Flip-Flop Fundamentals

T Flip-Flop Fundamentals

The T flip-flop (Toggle flip-flop) gets its name from its ability to toggle its output state. It has a single input T that controls its behavior:

- When T=1, the output toggles (changes state) on each clock pulse
- When T=0, the output maintains its current state with no change

The characteristic equation that defines a T flip-flop's behavior is:

$$Q_{n+1} = T'Q_n + TQ'_n \tag{1}$$

Where Q_n is the present state, Q_{n+1} is the next state, and T is the toggle input.

1.2 Converting JK Flip-Flop to T Flip-Flop

JK to T Flip-Flop Conversion

Since dedicated T flip-flop ICs are not commonly available, JK flip-flops are used to implement them. The conversion is straightforward:

- Connect both J and K inputs to the same signal source (T input)
- When J=K=0, the flip-flop holds its current state
- When J=K=1, the flip-flop toggles on each clock pulse

This connection arrangement makes the JK flip-flop behave exactly like a T flip-flop.

2 Circuit Design

2.1 Circuit Connections

Circuit Connections

The complete circuit connections for the MOD-7 asynchronous counter are shown in Table 1.

ccc

Table 1: Connections for Mod-7 Asynchronous Counter with Display

Component	Pin	Connection
Arduino Connections		
Arduino	Pin 13	Clock Input to First 7476 Flip-Flop
7476 (First Flip-Flop - Q1)		
7476 (IC1)	VCC (Pin 16)	+5V
7476 (IC1)	GND (Pin 8)	0V (Ground)
7476 (IC1)	J1, K1	+5V (HIGH)
7476 (IC1)	CLK1	Arduino Pin 13 (Clock)
7476 (IC1)	Q1 (Pin 15)	Clock for Second Flip-Flop
7476 (IC1)	PRE1, CLR1	+5V (HIGH)
7476 (IC1)	CLR1	Input from 7410 NAND Gate
7476 (Second Flip-Flop - Q2)		
7476 (IC2)	VCC (Pin 16)	+5V
7476 (IC2)	GND (Pin 8)	0V (Ground)
7476 (IC2)	J2, K2	+5V (HIGH)
7476 (IC2)	CLK2	Q0 Output from First Flip-Flop
7476 (IC2)	Q2 (Pin 15)	Input to 7410 NAND Gate
7476 (IC2)	PRE2	+5V (HIGH)
7476 (IC2)	CLR2	Input from 7410 NAND Gate
7476 (Third Flip-Flop - Q3)		
7476 (IC3)	VCC (Pin 16)	+5V
7476 (IC3)	GND (Pin 8)	0V (Ground)
7476 (IC3)	J3, K3	+5V (HIGH)
7476 (IC3)	CLK3	Q1 Output from Second Flip-Flop

Continued on next page

Continued from previous page		
Component	**Pin**	**Connection**
7476 (IC3)	Q3 (Pin 15)	Input to 7410 NAND Gate
7476 (IC3)	PRE3	+5V (HIGH)
7476 (IC3)	CLR3	Input from 7410 NAND Gate
7410 (NAND Gate for Reset)		
7410 (IC4)	VCC (Pin 14)	+5V
7410 (IC4)	GND (Pin 7)	0V (Ground)
7410 (IC4)	Input 1	Q0 from First 7476
7410 (IC4)	Input 2	Q1 from Second 7476
7410 (IC4)	Input 3	Q2 from Third 7476
7410 (IC4)	Output	CLR of All 7476 Flip-Flops
7447 (BCD to 7-Segment Decoder)		
7447 (IC5)	VCC (Pin 16)	+5V
7447 (IC5)	GND (Pin 8)	0V (Ground)
7447 (IC5)	A (Pin 7)	Q0 from First Flip-Flop
7447 (IC5)	B (Pin 1)	Q1 from Second Flip-Flop
7447 (IC5)	C (Pin 2)	Q2 from Third Flip-Flop
7447 (IC5)	D (Pin 6)	GND (Always 0 for MOD-7 Counter)
7447 (IC5)	Outputs (a-g)	Corresponding Pins of 7-Segment Display
7-Segment Display		
7-Segment	a	Output a from 7447
7-Segment	b	Output b from 7447
7-Segment	c	Output c from 7447
7-Segment	d	Output d from 7447
7-Segment	e	Output e from 7447
7-Segment	f	Output f from 7447
7-Segment	g	Output g from 7447
7-Segment	Common Anode	+5V via 220Ω Resistor

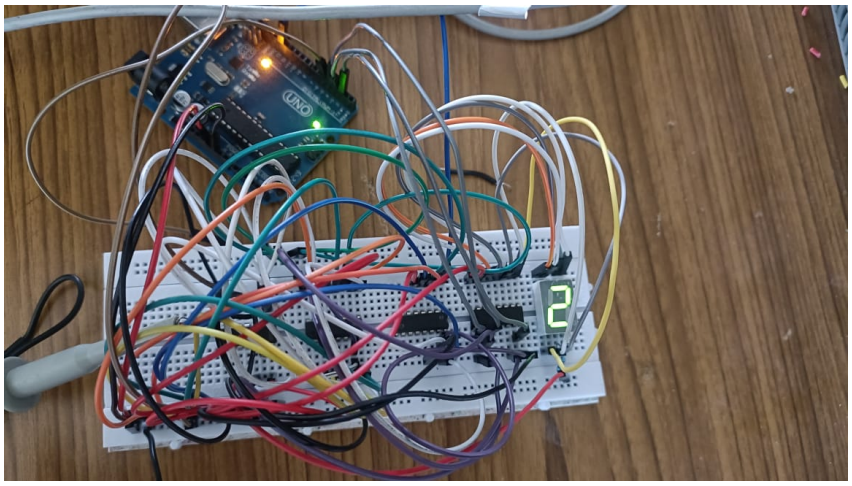

Figure 1: Circuit Diagram of Mod-7 Counter

2.2 T Flip-Flop Implementation

T Flip-Flop Implementation

Each JK flip-flop is configured as a T flip-flop by:

- Connecting both J and K inputs to logical HIGH (VCC)
- This makes them function in ”toggle mode” on each clock pulse
- The clock input of the first flip-flop comes from the Arduino
- The clock inputs of subsequent flip-flops come from the previous flip-flop’s output

2.3 Counter Architecture

Counter Architecture

The asynchronous MOD-7 counter consists of:

- Three JK flip-flops (configured as T flip-flops) forming a binary counter chain
- A reset detection circuit using a 3-input NAND gate
- A display decoder and 7-segment display

The counter operates as follows:

- The first flip-flop (Q0) receives the clock from Arduino and toggles on each clock pulse
- Q0 output clocks the second flip-flop (Q1), which toggles when Q0 transitions from HIGH to LOW
- Q1 output clocks the third flip-flop (Q2), which toggles when Q1 transitions from HIGH to LOW
- The NAND gate monitors all three outputs and resets the counter when it would reach state ”111”

3 Truth Tables and State Diagrams

3.1 JK Flip-Flop Truth Table

JK Flip-Flop Truth Table

J	K	Q(n)	Q(n+1)	Operation
0	0	0	0	No change
0	0	1	1	No change
0	1	0	0	Reset
0	1	1	0	Reset
1	0	0	1	Set
1	0	1	1	Set
1	1	0	1	Toggle
1	1	1	0	Toggle

3.2 MOD-7 Counter State Transition Table

MOD-7 Counter State Transition Table

Clock Cycle	Q2	Q1	Q0	Decimal	Action
0	0	0	0	0	Initial state
1	0	0	1	1	Q0 toggles
2	0	1	0	2	Q1 toggles, Q0 toggles
3	0	1	1	3	Q0 toggles
4	1	0	0	4	Q2 toggles, Q1 toggles, Q0 toggles
5	1	0	1	5	Q0 toggles
6	1	1	0	6	Q1 toggles, Q0 toggles
7	0	0	0	0	Reset occurs instead of 111

4 Observations and Results

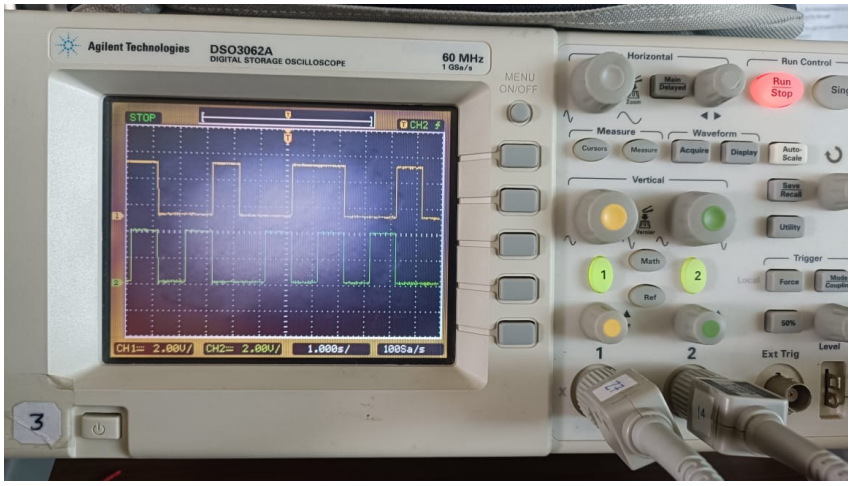

Figure 2: Channel1-Q2,Channel2-Q1

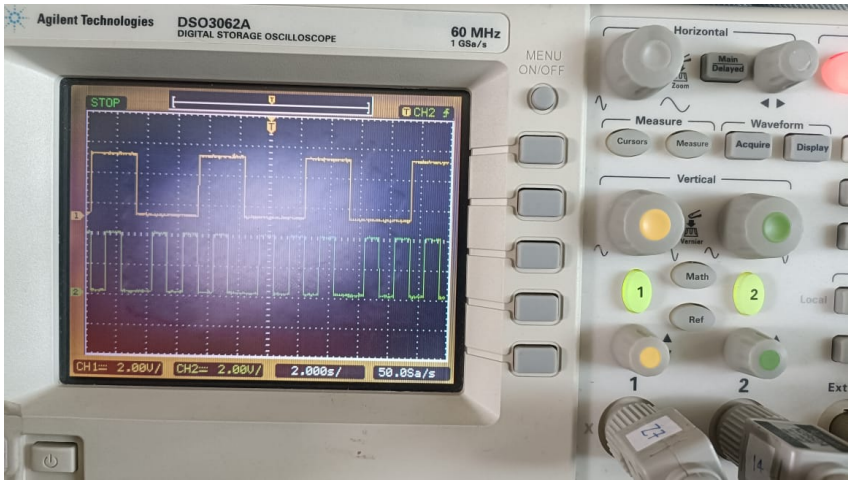

Figure 3: Channel1-Q3,Channel2-Q1

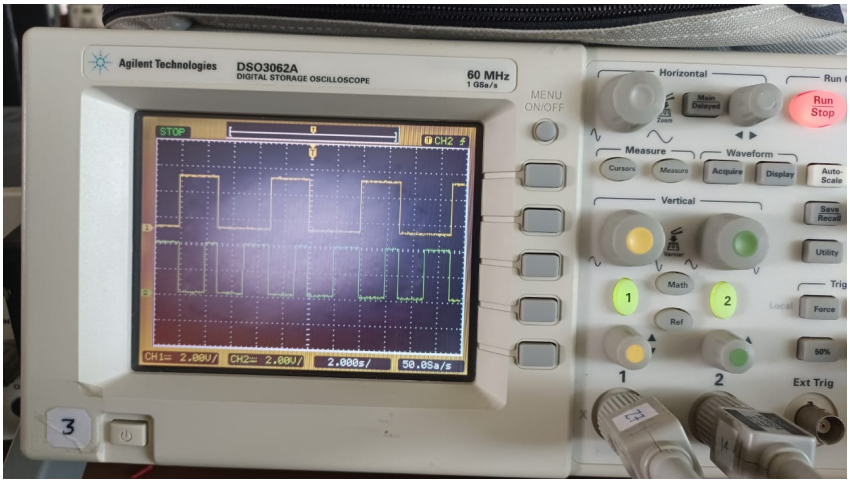

Figure 4: Channel1-Q3,Channel2-Q2

5 Arduino Clock Generation

Arduino Clock Generation Code

```
/*
 * MOD-7 Counter Clock Generator
 * This code generates a square wave clock signal for the MOD-7 counter
 * The frequency can be adjusted by changing the CLOCK_DELAY_MS constant
 */

const int CLOCK_PIN = 13;         // Clock output pin
const int CLOCK_DELAY_MS = 500;   // Half-period of clock in milliseconds
unsigned long clockCount = 0;     // Counter for clock pulses

void setup() {
  // Initialize serial communication for monitoring
  Serial.begin(9600);
  Serial.println("MOD-7 Counter Clock Generator");
  Serial.print("Clock Frequency: ");
  Serial.print(1000.0 / (2 * CLOCK_DELAY_MS));
  Serial.println(" Hz");

  // Configure clock pin as output
  pinMode(CLOCK_PIN, OUTPUT);
  digitalWrite(CLOCK_PIN, LOW);
}

void loop() {
  // Generate HIGH phase of clock
  digitalWrite(CLOCK_PIN, HIGH);
  Serial.print("Clock: HIGH | Count: ");
  Serial.println(clockCount);
  delay(CLOCK_DELAY_MS);

  // Generate LOW phase of clock
  digitalWrite(CLOCK_PIN, LOW);
  Serial.print("Clock: LOW  | Count: ");
  Serial.println(clockCount);
  delay(CLOCK_DELAY_MS);

  // Increment clock count for tracking
  clockCount++;
```

```
39
40    // Reset counter visualization after reaching 6
41    if (clockCount > 6) {
42      clockCount = 0;
43      Serial.println("-------- Counter Reset --------");
44    }
45  }
```

6 Conclusion

Conclusion

The MOD-7 asynchronous counter using T flip-flops (implemented with JK flip-flops) demonstrates several important concepts in digital electronics:

- The practical implementation of T flip-flops using more common JK flip-flops
- The design and operation of asynchronous (ripple) counters
- The modification of a natural binary counter to a modulo-n counter using reset logic
- The ripple effect and associated timing considerations in asynchronous designs

The counter successfully counts from 0 to 6 before resetting to 0, driven by a clock signal generated by an Arduino. The oscilloscope measurements confirm the expected behavior, including the frequency division at each stage and the reset action.
While asynchronous counters are simple to design and implement, their timing limitations make them less suitable for high-speed applications. For higher frequencies, synchronous counter designs would be preferred to eliminate the ripple delay issues.