

# EE LAB REPORT-7

## Authors:

EE24BTECH11033-KOLLURU SURAJ

EE24BTECH11038-M.B.S ARAVIND

logo.jpg

## CONTENTS

-A	IC's used . . . . .	3
-B	Logic of up counter . . . . .	3
<b>I</b>	<b>Karnaugh Maps for T Flip-Flop Inputs</b>	<b>4</b>
I-A	K-map for $T_A$ . . . . .	4
I-B	K-map for $T_B$ . . . . .	4
I-C	K-map for $T_C$ . . . . .	5
I-D	K-map for $T_D$ . . . . .	5
I-E	Logic of down counter . . . . .	5
<b>II</b>	<b>Karnaugh Maps for T Flip-Flop Inputs</b>	<b>6</b>
II-A	K-map for $T_A$ . . . . .	6
II-B	K-map for $T_B$ . . . . .	6
II-C	K-map for $T_C$ . . . . .	7
II-D	K-map for $T_D$ . . . . .	7
II-E	Clock signal generation for 1's place . . . . .	7
II-F	Clock signal generation for 10's place . . . . .	7
<b>III</b>	<b>Mod-7 Asynchronous Counter using JK Flip-Flops</b>	<b>7</b>
III-A	Introduction . . . . .	7
III-B	Apparatus . . . . .	8
III-C	Circuit Diagram . . . . .	9
III-D	Conversion of JK to T . . . . .	9
III-E	Theory / Working Principle . . . . .	10
	III-E1 Asynchronous counter . . . . .	10
	III-E2 Clock signal . . . . .	10
III-F	Truth Table . . . . .	11
III-G	Frequency Division in Asynchronous Counter . . . . .	11
<b>IV</b>	<b>Results</b>	<b>11</b>
	IV-1 First Flip-Flop (FF1) . . . . .	11
	IV-2 Second Flip-Flop (FF2) . . . . .	11
	IV-3 Third Flip-Flop (FF3) . . . . .	11
	IV-4 Time Period Analysis . . . . .	12
IV-A	Experimental Verification using LED's . . . . .	14
	IV-A1 Setup and Observations . . . . .	14
IV-B	Conclusion . . . . .	19

### *A. IC's used*

#### *74LS47 – BCD to 7-Segment Decoder/Driver:*

- Converts 4-bit Binary Coded Decimal (BCD) inputs to outputs for 7-segment displays.
- Designed to drive common anode displays with active-low outputs.
- Features include lamp test, blanking input, and ripple-blanking input/output.
- Can drive LED segments directly.

#### *74LS76 – Dual JK Flip-Flop with Clear and Preset:*

- Contains two independent JK flip-flops.
- Each flip-flop has inputs: J, K, Clock, Clear, and Preset.
- Supports set, reset, toggle, and hold operations.
- Edge-triggered; suitable for use in counters and registers.

#### *74LS32 – Quad 2-Input OR Gate:*

- Includes four independent 2-input OR gates.
- Logical OR operation: output is HIGH if any input is HIGH.
- Commonly used in decision-making logic circuits.
- Fast switching with low power consumption.

#### *74LS11 – Triple 3-Input AND Gate:*

- Contains three independent 3-input AND gates.
- Output is HIGH only when all three inputs are HIGH.
- Used for implementing multiple-input logic conditions.
- Standard TTL-compatible inputs and outputs.

### *B. Logic of up counter*

The state table for the MOD-10 up counter is given below:

Present State				Next State				T Flip-Flop Inputs
$Q_A$	$Q_B$	$Q_C$	$Q_D$	$Q_{A+1}$	$Q_{B+1}$	$Q_{C+1}$	$Q_{D+1}$	$T_A, T_B, T_C, T_D$
0	0	0	0	0	0	0	1	0 0 0 1
0	0	0	1	0	0	1	0	0 0 1 1
0	0	1	0	0	0	1	1	0 0 0 1
0	0	1	1	0	1	0	0	0 1 1 1
0	1	0	0	0	1	0	1	0 0 0 1
0	1	0	1	0	1	1	0	0 0 1 1
0	1	1	0	0	1	1	1	0 0 0 1
0	1	1	1	1	0	0	0	1 1 1 1
1	0	0	0	1	0	0	1	0 0 0 1
1	0	0	1	0	0	0	0	1 0 0 1
1	0	1	0	X	X	X	X	X X X X
1	0	1	1	X	X	X	X	X X X X
1	1	0	0	X	X	X	X	X X X X
1	1	0	1	X	X	X	X	X X X X
1	1	1	0	X	X	X	X	X X X X
1	1	1	1	X	X	X	X	X X X X

### I. KARNAUGH MAPS FOR T FLIP-FLOP INPUTS

The required flip-flop excitations are derived using Karnaugh maps:

#### A. K-map for $T_A$

logo.jpg

$$T_A = Q_A Q_D + Q_B Q_C Q_D$$

(1)

$Q_C Q_D$	00	01	11	10
$Q_A Q_B$				
00				
01			1	
11	X	X	X	X
10		1	X	X

#### B. K-map for $T_B$

$$T_B = Q_C Q_D$$

(2)

$Q_A Q_B \backslash Q_C Q_D$	00	01	11	10
00			1	
01			1	
11	X	X	X	X
10			X	X

C. K-map for  $T_C$

$$T_C = \overline{Q_A} Q_D$$

(3)

$Q_A Q_B \backslash Q_C Q_D$	00	01	11	10
00		1	1	
01		1	1	
11	X	X	X	X
10			X	X

D. K-map for  $T_D$

$$T_D = 1$$

(4)

$Q_A Q_B \backslash Q_C Q_D$	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

E. Logic of down counter

The state table for the MOD-10 down counter is given below:

Present State				Next State				T Flip-Flop Inputs
$Q_A$	$Q_B$	$Q_C$	$Q_D$	$Q_{A+1}$	$Q_{B+1}$	$Q_{C+1}$	$Q_{D+1}$	$T_A, T_B, T_C, T_D$
1	0	0	1	1	0	0	0	0 0 0 1
1	0	0	0	0	1	1	1	1 1 1 1
0	1	1	1	0	1	1	0	0 0 0 1
0	1	1	0	0	1	0	1	0 0 1 1
0	1	0	1	0	1	0	0	0 0 0 1
0	1	0	0	0	0	1	1	0 1 1 1
0	0	1	1	0	0	1	0	0 0 0 1
0	0	1	0	0	0	0	1	0 0 1 1
0	0	0	1	0	0	0	0	0 0 0 1
0	0	0	0	1	0	0	1	1 0 0 1
1	0	1	0	X	X	X	X	X X X X
1	0	1	1	X	X	X	X	X X X X
1	1	0	0	X	X	X	X	X X X X
1	1	0	1	X	X	X	X	X X X X
1	1	1	0	X	X	X	X	X X X X
1	1	1	1	X	X	X	X	X X X X

## II. KARNAUGH MAPS FOR T FLIP-FLOP INPUTS

The required flip-flop excitations are derived using Karnaugh maps:

### A. K-map for $T_A$

logo.jpg

$$T_A = \overline{Q_B Q_C Q_D}$$

(5)

$Q_C Q_D \backslash Q_A Q_B$	00	01	11	10
00	1			
01				
11	X	X	X	X
10	1		X	X

### B. K-map for $T_B$

$$T_B = Q_B \overline{Q_C Q_D} + Q_A \overline{Q_C Q_D}$$

(6)

$Q_C Q_D \backslash Q_A Q_B$	00	01	11	10
00				
01	1			
11	X	X	X	X
10	1		X	X

### C. K-map for $T_C$

$$T_C = Q_C \overline{Q_D} + Q_A \overline{Q_D} + Q_B \overline{Q_D} \quad (7)$$

$Q_C Q_D \backslash Q_A Q_B$	00	01	11	10
00				1
01	1			1
11	X	X	X	X
10	1		X	X

### D. K-map for $T_D$

$$T_D = 1 \quad (8)$$

### E. Clock signal generation for 1's place

The one's place clock signal is generated using an Arduino. A push button is connected to the Arduino, and each time it is pressed, a clock pulse is triggered programmatically. This method allows manual control over the clock input of the one's place T flip-flops.

#### ARDUINO CODE

### F. Clock signal generation for 10's place

- 1) For the ten's place clock signal, use two AND gates. For the increment operation, connect the increment button and the output  $Q_3 Q_0$  of the one's place to an AND gate. For the decrement operation, connect the decrement button and the complemented outputs  $\overline{Q_0 Q_1 Q_2 Q_3}$  to another AND gate. Then, OR the outputs of both AND gates to generate the final clock signal for the ten's place counter.

## III. MOD-7 ASYNCHRONOUS COUNTER USING JK FLIP-FLOPS

### A. Introduction

In this experiment, a Mod-7 asynchronous counter was built using JK Flip-Flops configured as T Flip-Flops by setting J and K to the same logic level. The counter operates on the falling edge of the clock

and resets using a NAND gate when the count reaches  $111_2$  (7). An Arduino-generated clock was used to drive the circuit, and LEDs were used to display the count. The output frequency of the last flip-flop was verified as  $f/7$ , confirming correct operation.

**The pin layout of the 74LS76 JK Flip-Flop is shown below**



Fig. 1: JK flip-flop(74LS76)

The 74LS76 is a dual JK flip-flop with Preset (PRE) and Clear (CLR) inputs, which are asynchronous active-low signals. Each flip-flop has J, K, Clock (CLK), Q, and  $\bar{Q}$  outputs, allowing versatile configurations such as T flip-flops. The pin layout helps in understanding the proper wiring for sequential logic applications like counters and frequency dividers.

**The pin layout of 74LS10 NAND gate is shown below**

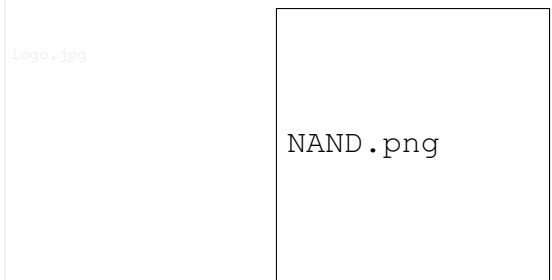


Fig. 2: NAND gate

The 74LS10 is a triple 3-input NAND gate IC, meaning it contains three independent NAND gates, each with three inputs. The pin layout defines the connections for inputs (A, B, C) and outputs (Y) for each gate, enabling its use in various logic operations, including combinational and sequential circuits.

### *B. Apparatus*

The following components were used to implement the Mod-7 asynchronous counter:

- **JK Flip-Flops (3x)** – Configured as T Flip-Flops by setting  $J = K$
- **NAND Gate (1x)** – For reset logic when the count reaches  $111_2$  (7)



- **LEDs (3x)** – To visually display the binary count
- **Resistors (appropriate values)** – To limit current for LEDs
- **Arduino Uno** – To generate the clock signal
- **Oscilloscope (CRO)** – To observe the clock and output waveforms
- **Breadboard and Jumper Wires** – For circuit assembly

### C. Circuit Diagram



Fig. 3: Circuit Diagram of Counter

### D. Conversion of JK to T

A T flip-flop has a single input (T) and toggles its state on every clock pulse when  $T = 1$ , while maintaining its previous state when  $T = 0$ . The JK flip-flop, on the other hand, has two inputs (J and K) and follows this characteristic:

J	K	Next State ( $Q_{n+1}$ )
0	0	No Change
0	1	Reset ( $Q = 0$ )
1	0	Set ( $Q = 1$ )
1	1	Toggle ( $\bar{Q}$ )

To achieve T flip-flop functionality, we set  $J = T$  and  $K = T$ , ensuring that:

- When  $T = 0$ , the flip-flop holds its state.
- When  $T = 1$ , the flip-flop toggles on every clock pulse.

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

### E. Theory / Working Principle

1) *Asynchronous counter:* An asynchronous counter is a type of counter where flip-flops are not clocked simultaneously; instead, the clock pulse propagates through the flip-flops sequentially. This causes a ripple effect, leading to propagation delays. While simpler in design, asynchronous counters are slower compared to synchronous counters due to these delays.

2) *Clock signal:* This Arduino code generates a clock signal on pins 2 and 3, which can be used to drive digital circuits like flip-flops and counters. The signal alternates between HIGH and LOW with a 500-ms delay, creating a periodic waveform for synchronization.

Listing 1: Arduino Code

```

1 #define CLOCK_PIN 2
2 #define CL 3
3
4 void setup() {
5     pinMode(CLOCK_PIN, OUTPUT);
6     pinMode(CL, OUTPUT);
7 }
8
9 void loop() {
10    digitalWrite(CL, HIGH);
11    digitalWrite(CLOCK_PIN, HIGH);
12    delay(500); // Adjust for desired frequency
13    digitalWrite(CL, HIGH);
14    digitalWrite(CLOCK_PIN, LOW);
15    delay(500);
16 }
```

A **Mod-7 asynchronous counter** is a sequential circuit that cycles through seven states (0 to 6) using **JK Flip-Flops configured as T Flip-Flops**. In this configuration, the J and K inputs are tied together and set to logic HIGH, which creates a toggle mode equivalent to a T Flip-Flop. Each Flip-Flop toggles its state on the falling edge of the clock signal, causing a binary count sequence.

When  $J = K = 1$ , the JK Flip-Flop toggles its state on each clock pulse, effectively functioning as a T Flip-Flop. The asynchronous nature means the output of one Flip-Flop serves as the clock input for the next, creating a ripple effect. The frequency of the last Flip-Flop output is divided by 7 relative to the input clock.

The three Flip-Flops ( $JK_0$ ,  $JK_1$ , and  $JK_2$ ) represent the least significant bit (LSB) to the most significant bit (MSB) of the count. The first Flip-Flop ( $JK_0$ ) toggles on every falling edge of the external clock. The second Flip-Flop ( $JK_1$ ) toggles when  $JK_0$  transitions from high to low, and the third Flip-Flop ( $JK_2$ ) toggles when  $JK_1$  transitions. To ensure the counter resets after reaching 6, a **NAND gate** detects the count '111<sub>2</sub>' (decimal 7) and asynchronously clears all Flip-Flops, bringing the count back to 000.

The PRE and CLR of flip flops are set to be 1 for proper working, but here we need to reset when we get out put  $(111)_2 = 7$  . When CLR is 0 the output is forced to be zero we will use this here

So we will use a NAND gate to make it happen, all the outputs of Flips flops are connected to the NAND gate and the outputs of NAND gate are connected to the CLR of all Flip flops so that when 111 is the input the output is 0 and all CLR's will be set to 0 so it will start from 0.

#### F. Truth Table

The following truth table represents the binary count sequence of the Mod-7 asynchronous counter:

Count (Decimal)	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
(Reset)	0	0	0

TABLE I: Truth Table of the Mod-7 Asynchronous Counter

#### G. Frequency Division in Asynchronous Counter

In an asynchronous counter without a reset mechanism(Here a NAND Gate), each successive flip-flop effectively divides the frequency of the previous stage by 2. The first flip-flop toggles at  $f/2$  of the input clock, the second flip-flop toggles at  $f/4$ , and the third flip-flop toggles at  $f/8$ . This natural frequency division occurs because each flip-flop changes state only when the previous stage transitions from high to low, creating a cascading effect that progressively reduces the output frequency. The binary outputs (Q<sub>0</sub>, Q<sub>1</sub>, Q<sub>2</sub>) represent these divided frequencies, with each bit representing a power of 2 reduction from the original clock signal.

### IV. RESULTS

#### Without NAND gate

1) *First Flip-Flop (FF1)*: The first flip-flop (FF1) toggles once every clock pulse, so its output has a frequency:

$$f_{Q_0} = \frac{f_{clk}}{2} \quad (9)$$

2) *Second Flip-Flop (FF2)*: The second flip-flop toggles whenever transitions from 1 to 0 of Q<sub>0</sub>, which occurs every two clock pulses:

$$f_{Q_1} = \frac{f_{Q_0}}{2} = \frac{f_{clk}}{4} \quad (10)$$

3) *Third Flip-Flop (FF3)*: The third flip-flop toggles on the falling edge of second Flip flop Q<sub>1</sub>, occurring once every four clock pulses:

$$f_{Q_2} = \frac{f_{Q_1}}{2} = \frac{f_{clk}}{8} \quad (11)$$

In general, for an n-bit asynchronous counter, the frequency of the th flip-flop is given by:

$$f_{Q_n} = \frac{f_{clk}}{2^{(n+1)}} \quad (12)$$

where starts from 0 for the first flip-flop.

4) *Time Period Analysis*: The clock input has a time period:

$$T_{clk} = \frac{1}{f_{clk}} \quad (13)$$

For each output:

$$T_{Q_0} = 2T_{clk} \Rightarrow f_{Q_0} = \frac{1}{2T_{clk}} = \frac{f_{clk}}{2} \quad (14)$$

$$T_{Q_1} = 2T_{Q_0} = 4T_{clk} \Rightarrow f_{Q_1} = \frac{1}{4T_{clk}} = \frac{f_{clk}}{4} \quad (15)$$

$$T_{Q_2} = 2T_{Q_1} = 8T_{clk} \Rightarrow f_{Q_2} = \frac{1}{8T_{clk}} = \frac{f_{clk}}{8} \quad (16)$$

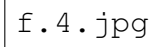
This confirms that, without using a NAND gate for resetting, the output frequencies follow a binary division sequence:

$$f_{Q_0} = \frac{f_{clk}}{2}, \quad f_{Q_1} = \frac{f_{clk}}{4}, \quad f_{Q_2} = \frac{f_{clk}}{8}, \quad \text{and so on.} \quad (17)$$

logo.jpg

f.2.jpg

Fig. 4: Response of  $Q_0$  and clock

A timing diagram showing the response of  $Q_1$  to a clock signal. The diagram is contained within a rectangular frame. The text "f.4.jpg" is visible in the top-left corner of the frame.

f.4.jpg

Fig. 5: Response of  $Q_1$  and clock

A timing diagram showing the response of  $Q_2$  to a clock signal. The diagram is contained within a rectangular frame. The text "logo.jpg" is visible in the top-left corner of the frame.

logo.jpg

f.8.jpg

Fig. 6: Response of  $Q_2$  and clock

**With NAND Gate**

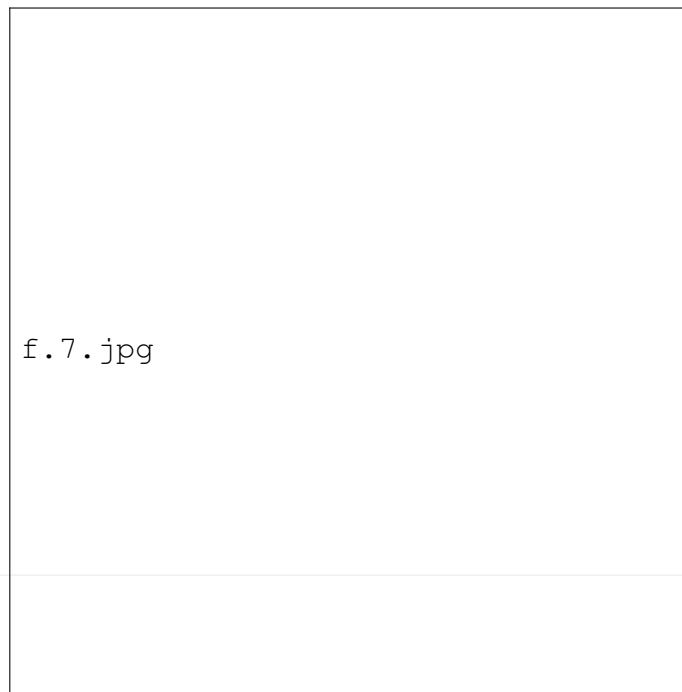


Fig. 7: Response of  $Q_2$  and clock

We can observe that the frequency is  $f/7$ . It is exact according to the truth table (more 0 and less 1)

#### A. Experimental Verification using LED's

To verify the behavior of the asynchronous counter with a NAND reset, an LED display was used to visualize the state transitions of the flip-flops. The experiment aimed to observe how the frequency of each output signal changed due to the NAND gate reset and to analyze the resulting LED blinking patterns.

##### 1) Setup and Observations:

- The outputs of each flip-flop in the counter circuit were connected to individual LEDs.
- An Arduino provided the clock signal to drive the counter.
- The LEDs toggled based on the counter's binary state, making it possible to observe the frequency division pattern.

000.png

logo.jpg

Fig. 8: Output of 000

010.png

logo.jpg

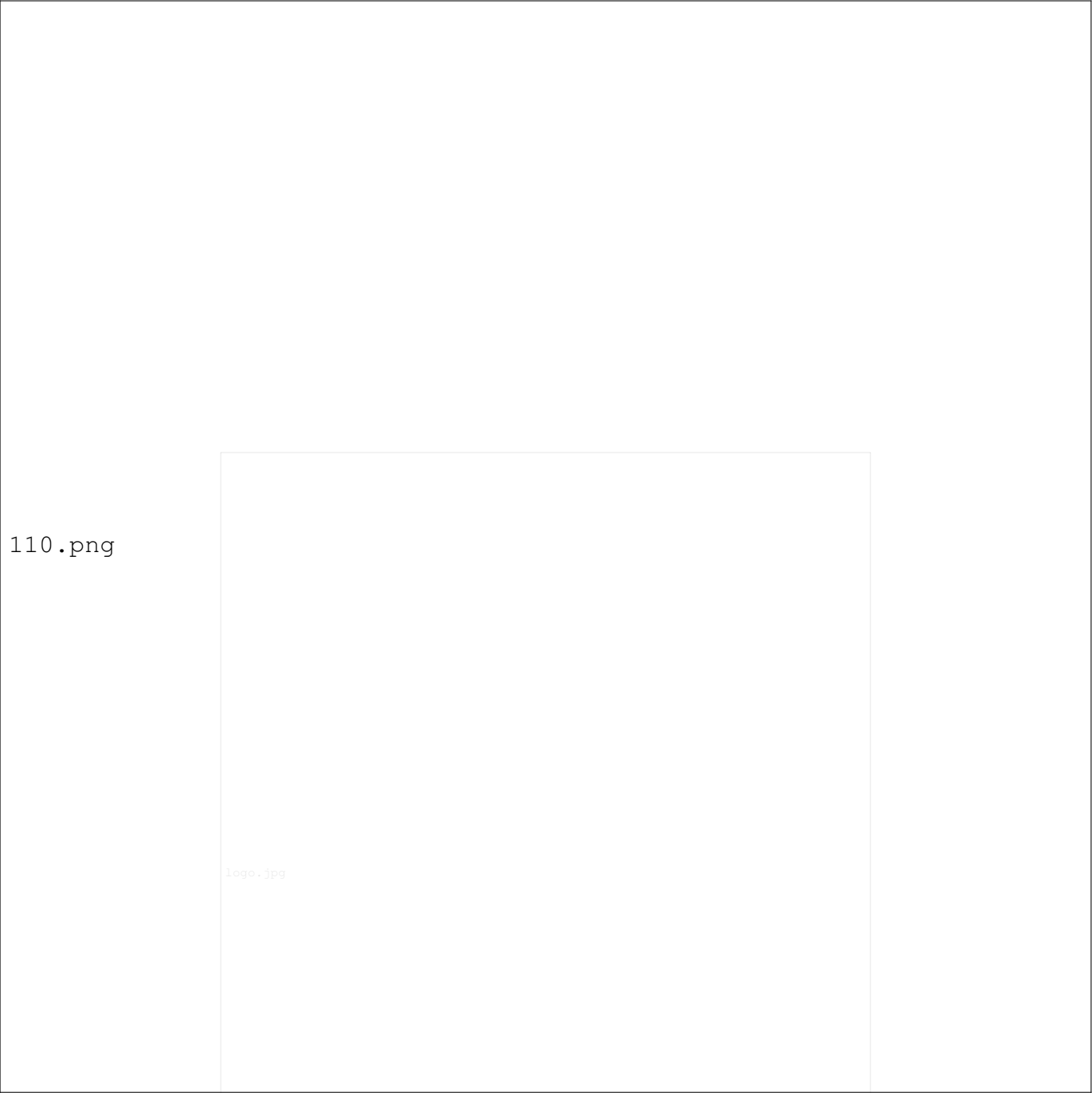
Fig. 10: Output of 010



100.png

logo.jpg

Fig. 12: Output of 100



110.png

logo.jpg

Fig. 14: Output of 110

## *B. Conclusion*

The experimental study of the asynchronous counter with and without a NAND reset provided a comprehensive understanding of its frequency division characteristics. Without the NAND reset, the flip-flops followed a standard binary sequence, producing frequencies of  $f/2$ ,  $f/4$ ,  $f/8$ , and so on. However, introducing a NAND gate to reset the counter at a predetermined count significantly altered the frequency behavior, yielding non-binary frequency divisions such as  $f/7$ ,  $f/3$ , etc.

The LED display method proved to be an effective tool for visualizing state transitions in real time. The observed LED blinking patterns aligned with theoretical predictions, confirming the impact of the NAND reset on frequency modulation. These findings emphasize the practical significance of Mod-N counters in frequency division applications, making them essential components in digital electronics and embedded systems.

Overall, the experiment successfully demonstrated how modifying the counter design with a NAND gate influences output frequency and cycle patterns. This knowledge is crucial for designing efficient digital circuits where precise timing and frequency control are required.



logo.jpg