CONTENTS

SYNOPSIS

Ensuring accurate taxi fares for long journeys is crucial. This synopsis explores a VLSI design for a fare meter that overcomes counter limitations. By utilizing a parallel adder and a predefined value during overflow, the system guarantees accurate fare calculation even for extended trips.

The taxi fare meter will consist of the following main components counter ,this component keeps track of the elapsed time during the journey. It can be a synchronous or asynchronous counter depending on the design preference.Parallel adder which efficiently calculates the total fare by adding the base fare with the product of time and fare rate.overflow detector which detects when the counter reaches its maximum value and generates an overflow signal. PIPO register stores the calculated fare for display and potential integration with a payment system. Control unit controls the overall operation of the system, including triggering fare calculation, handling overflow, and updating the display.

During regular operation, the counter increments with each time unit (e.g., second). The current counter value and a constant value representing the fare rate are fed into the parallel adder along with a base fare value. The adder calculates the total fare, and the result is loaded into the PIPO register for display. Upon receiving the overflow signal, the parallel adder adds the predefined value to the fare being calculated. This effectively extends the fare calculation beyond the counter's limit, incorporating the overflow into the total fare.

The Cadence tool suite provides a comprehensive environment for designing and simulating the taxi fare meter circuit. The design flow would involve creation of symbols for each component and connecting them according to the system design. The overflow signal path from the counter to the parallel adder with the predefined value input needs to be clearly defined. This part comes under schematic entry, then simulation and verification involves simulating the designed system using Cadence simulators, to verify its functionality under various time intervals, base fares, and overflow scenarios. Ensuring accurate fare calculation even when the counter overflows.

INTRODUCTION

Cadence:

Cadence Design Systems, Inc. (stylized as cādence), is a multinational computational software company, Headquartered in San Jose, California, Cadence was formed in 1988 through the merger of SDA Systems and ECAD. Initially specialized in electronic design automation (EDA) software for the semiconductor industry, currently the company makes software and hardware for designing products such as integrated circuits, systems on chips (SoCs), and printed circuit boards.

Schematic Entry: Cadence tools allow you to visually represent the system components using symbols and connect them according to the design. This provides a clear and intuitive way to build the circuit schematic for the taxi fare meter, including the counter, parallel adder, overflow detection logic, and PIPO register.

Simulation: Cadence simulators allow you to virtually test the designed circuit under various conditions. You can simulate different journey times, base fares, and overflow scenarios to ensure the taxi fare meter functions correctly and calculates accurate fares for all cases.

Comparison of Traditional Taxi Fare Meter with our VLSI Design:

Traditional taxi fare meters typically rely on the following components:

- **Counter:** This electronic circuit keeps track of the elapsed journey time. It increments (counts up) at regular intervals, often every second.
- **Fare Rate:** This is a predefined value representing the cost per unit time (e.g., per second or minute).
- **Base Fare:** This is a fixed fee charged at the beginning of the journey.
- **Display Unit:** This displays the current fare to the passenger.

Fare Calculation:

The fare is calculated by multiplying the elapsed time (counter value) by the fare rate and adding the base fare. However, there's a limitation:

- **Counter Overflow:** When the counter reaches its maximum value, it overflows and resets to zero. This poses a problem for long journeys exceeding the counter's capacity.

Our VLSI Design with Overflow Handling:

Our design addresses this limitation by introducing a novel approach:

- **Overflow Detection:** We incorporate a circuit that detects when the counter overflows.
- **Predefined Value:** We define a fixed value representing a reasonable extension of the fare calculation beyond the counter's limit.
- **Parallel Adder:** This component efficiently handles fare calculations.

Benefits:

- **Accurate Fare Calculation:** Even for extended journeys, the pre-defined value ensures the calculated fare reflects the entire trip duration, overcoming the counter overflow limitation.
- **Improved Reliability:** This design approach eliminates inaccurate fares due to counter overflows, leading to a more reliable and trustworthy taxi fare meter system.

Comparison:

Traditional fare meters can lead to inaccurate fares for long journeys exceeding the counter's limit. Our VLSI design with overflow handling provides a robust solution, guaranteeing accurate fare calculation regardless of trip duration.

DESIGN METHODOLOGY AND IMPLEMENTATION

Flowchart for the design of taxi meter:

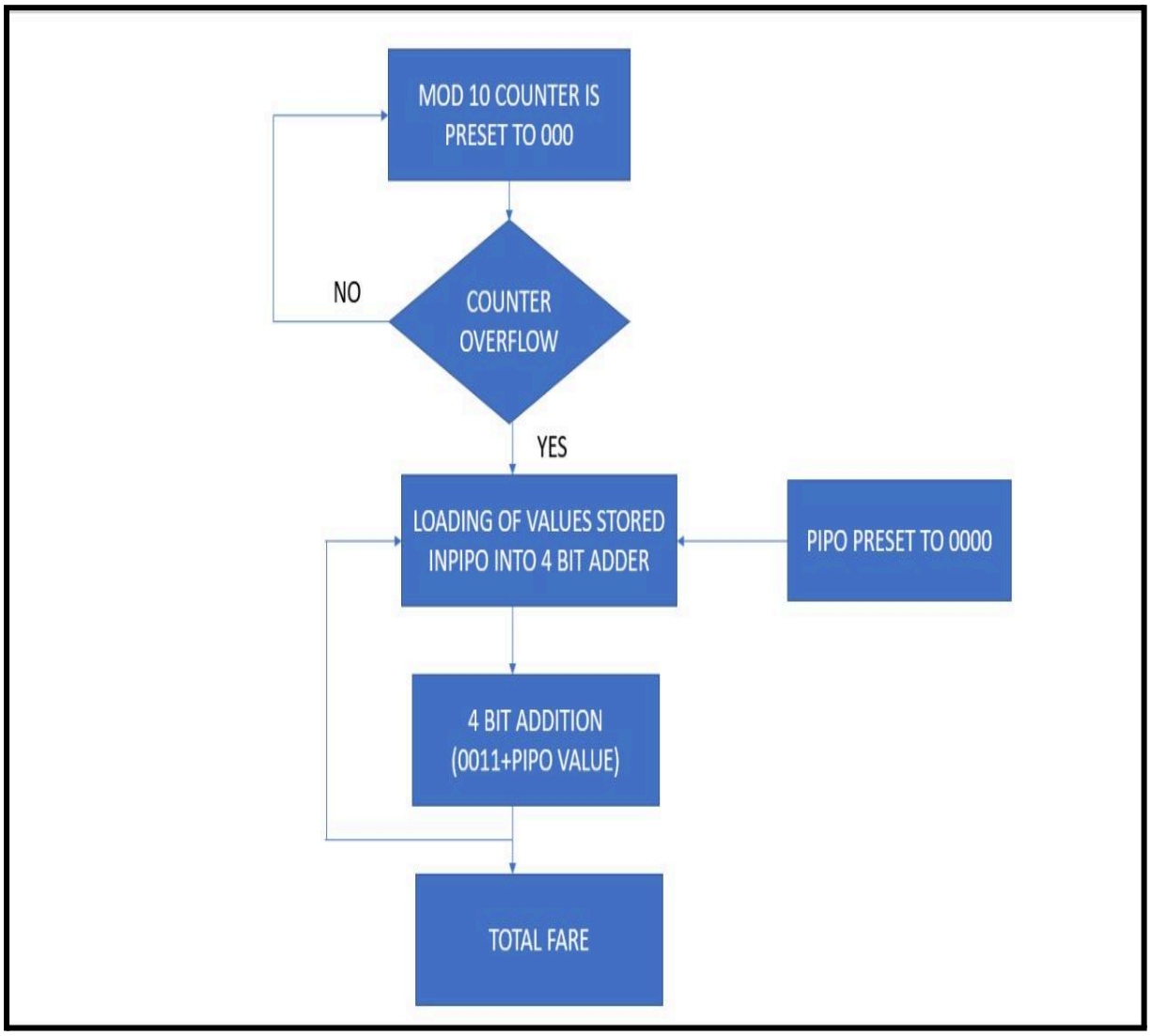

Fig 1: Flowchart

Components used in the circuit:

The circuit incorporates several key components: a 4-bit counter, a ripple carry adder, and a parallel-in parallel-out (PIPO) circuit. Within this setup, the 4-bit counter functions as a mod-10 counter, enabling it to count up to a maximum of 10. The PIPO circuit operates by simultaneously receiving inputs from the full adder. When the counter overflows, indicating it has reached its maximum count, the PIPO circuit adds these inputs to the next value. Additionally, the counter resets to zero after every kilometer, ensuring continuous and accurate counting cycles.

4-bit counter(mod-10):

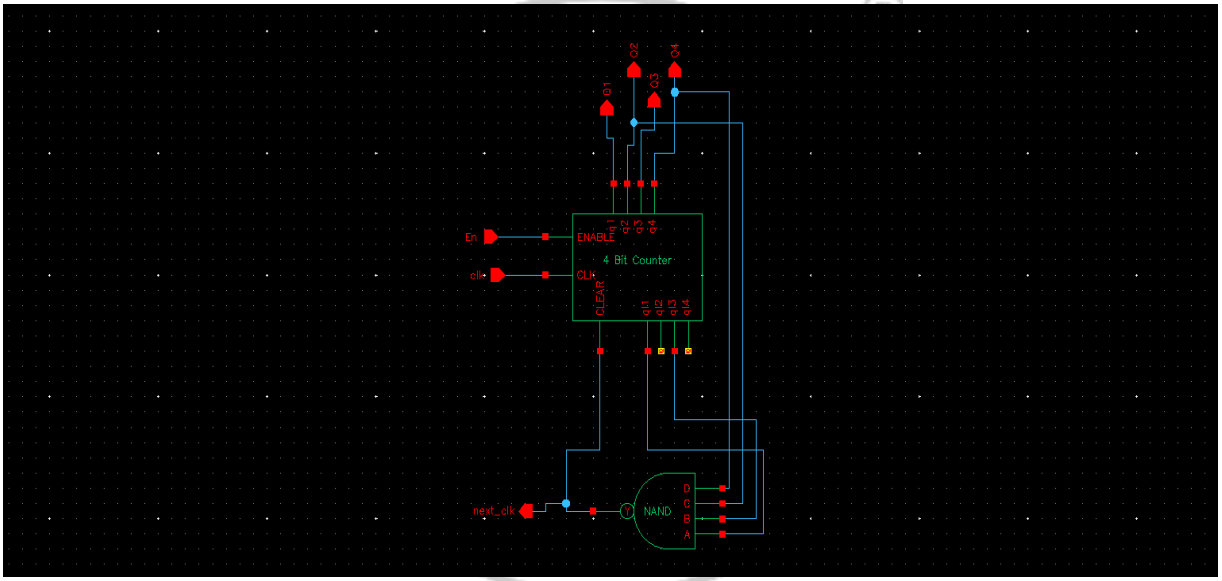

Fig 2: 4-bit mod-10 counter

Components:

- Flip-flops: Four D flip-flops (D0, D1, D2, and D3) are used to store the count value in binary format. Each flip-flop can be in either a 0 or 1 state.
- Logic gates: AND gates, OR gates, or a combination of these are used to implement the logic for counting and resetting.

Operation:

1. **Counting Sequence:**
    - The logic gates are designed to generate the next count value based on the current outputs of the flip-flops.
    - For example, an AND gate might be used to detect when the count reaches a specific condition (like D0 being low and D1 being high, representing binary 01).
    - When this condition is met, the output of the AND gate will set the input of the next flip-flop (D2 in this example) to high, causing it to toggle to 1 on the next clock pulse.
    - This ripple effect propagates through the flip-flops, generating the counting sequence from 0000 (binary 0) to 1001 (binary 9).
2. **Resetting the Counter:**
    - The logic gates also generate a signal to reset the counter when it reaches 1001 (binary 9). This might involve another AND gate configuration that detects this specific output combination.
    - The output of this reset logic will clear the data inputs (D) of all the flip-flops to 0, causing them all to reset to 0 on the next clock pulse. This brings the counter back to 0000 and the counting cycle restarts.

Symbol of mod-10 counter:

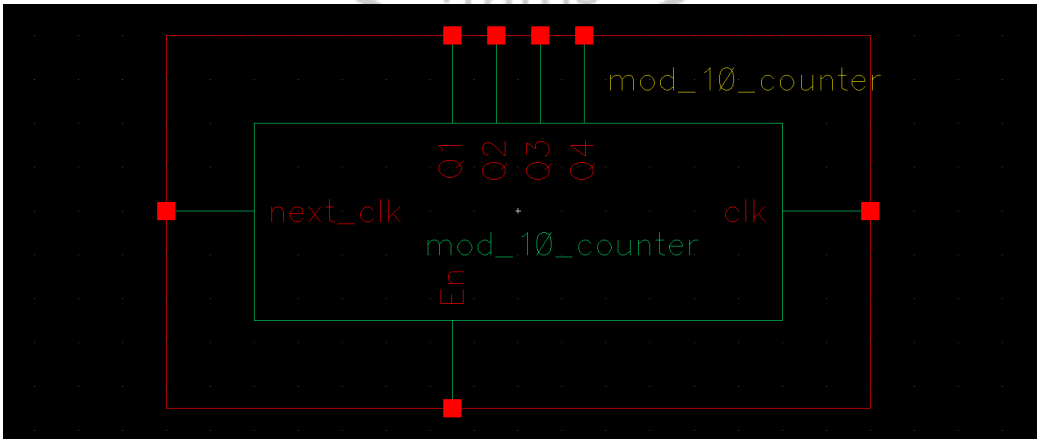

Fig 3: symbol of mod-10 counter

4-bit ripple carry adder:

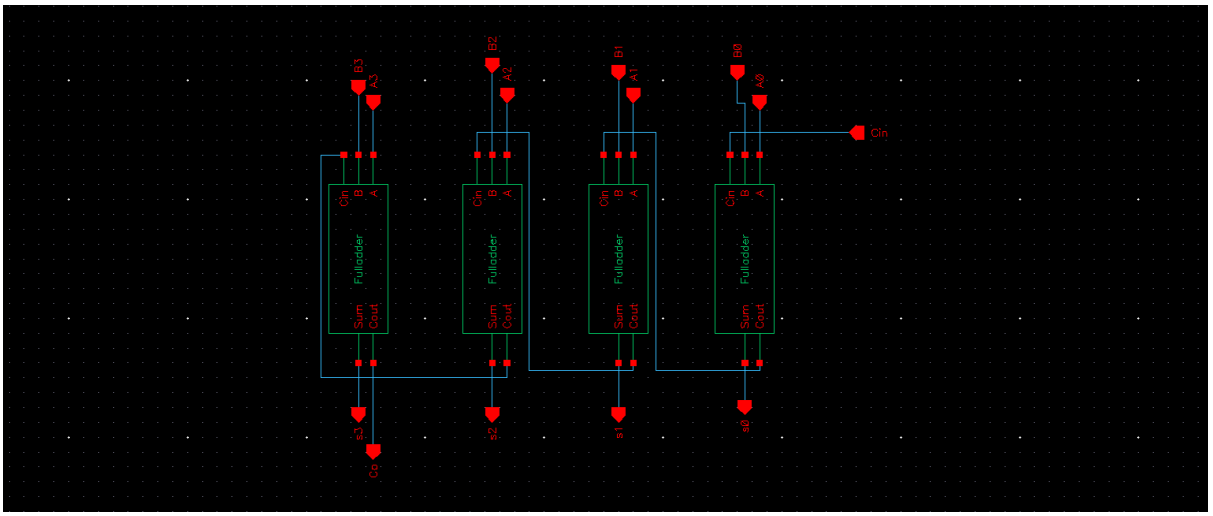

Fig 4: 4-bit ripple carry adder

Components:

- **Full Adders (FA):** Four full adders are the building blocks of the ripple carry adder. Each full adder takes three binary inputs (A, B, and Carry-in) and produces two binary outputs (Sum and Carry-out).
    - A and B are the two bits to be added.
    - Carry-in represents the carry bit from the previous less significant addition (explained later).
    - Sum represents the sum of A, B, and Carry-in.
    - Carry-out represents the carry bit generated from the current addition, which becomes the Carry-in for the next full adder.
- The four Sum bits (S0, S1, S2, S3) from the full adders represent the 4-bit binary sum of the two input numbers.
- The final Carry-out (from the fourth full adder) indicates if there was an overflow during the addition (resulting in a carry beyond the 4 bits).

Symbol for 4-bit ripple carry adder:

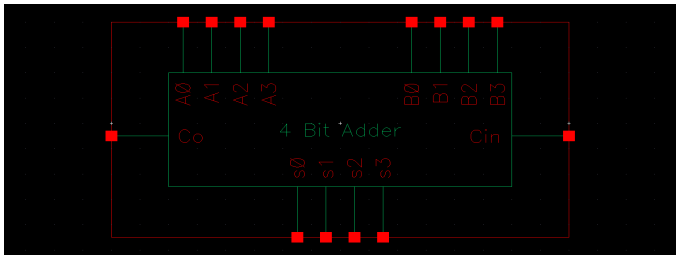

Fig 5: symbol of 4-bit ripple carry adder

Parallel In Parallel Out(PIPO):

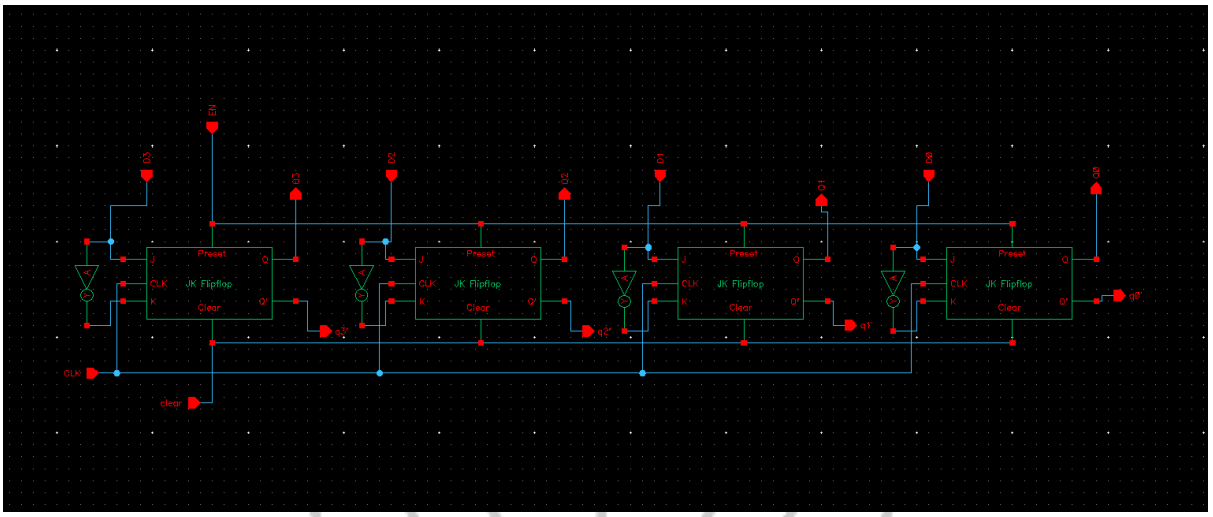

Fig 6: PIPO

Components:

- **JK Flip-Flops:** One JK flip-flop is needed for each data bit you want to store in the register. A JK flip-flop has two inputs (J and K) and two outputs (Q and $\overline{Q}$).
- **Control Logic:** Additional logic gates (usually AND gates or OR gates) are used to control the loading of data into the flip-flops.
- **Enable Input (Optional):** An enable input can be added to the control logic to allow loading of data only when the enable signal is active.

Operation:

1. **Data Loading:**
    - Each data bit to be loaded is connected to the J input of its corresponding JK

flip-flop. The K input of each flip-flop is typically tied to logic 0 (LOW) or logic 1 (HIGH) depending on the desired loading mechanism.

- A control signal is generated by the logic gates. This signal might be active HIGH (e.g., from an AND gate where all data bits are present) or active LOW (e.g., from an OR gate where any data bit is high).

2. **Clock Pulse:**
   - When the control signal is active and a clock pulse arrives, the JK flip-flops capture the data present at their J inputs. The data is then stored in the Q outputs of the flip-flops.
3. **Holding Data:**
   - As long as the clock doesn't change, the data remains stored in the Q outputs of the flip-flops, even if the control signal or data inputs change.
4. **Parallel Output:**
   - The Q outputs of all the flip-flops represent the stored data in parallel. You can connect them to other circuits to use the data.

Symbol for Parallel in Parallel Out:

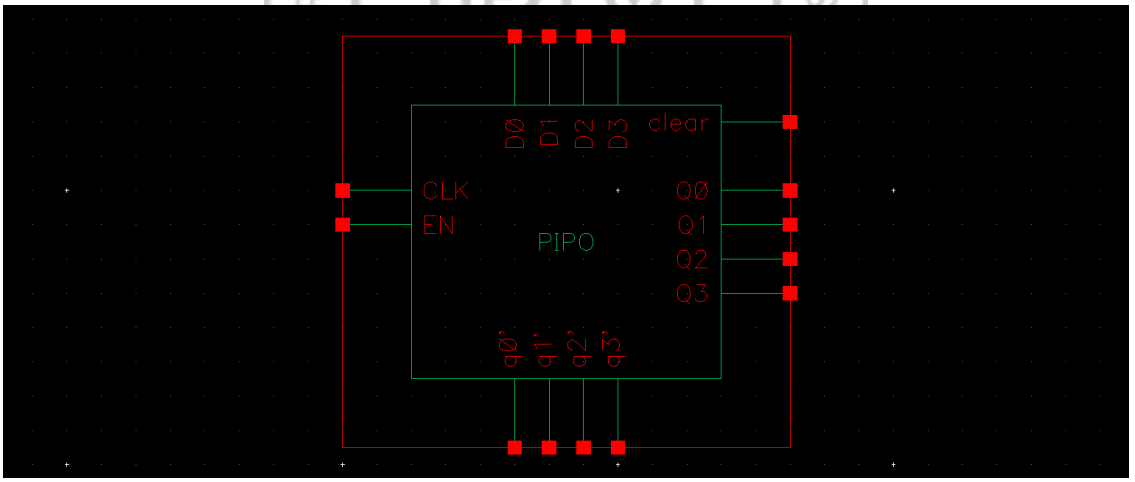

Fig 7: Symbol of PIPO

Design of Digital Taxi Fare Meter:

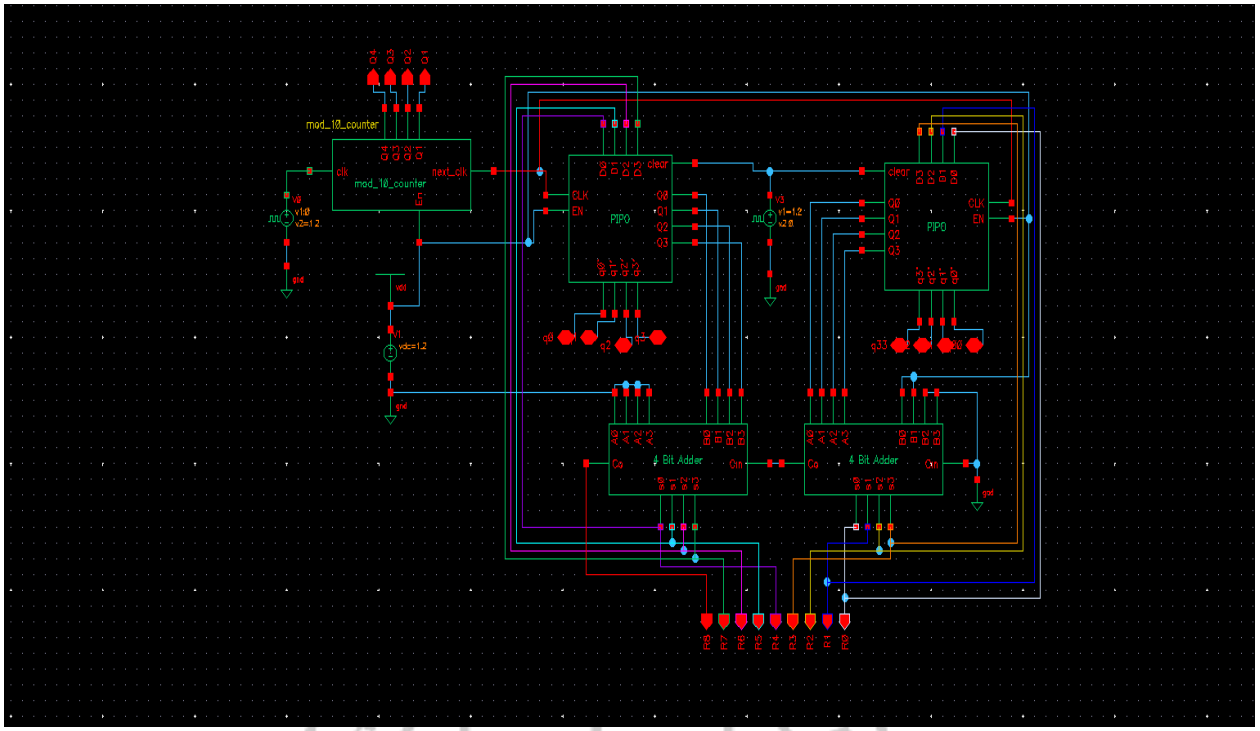

Fig 8: Taxi Fare Meter

A mod-10 counter is utilized to indicate kilometers, where each overflow of the counter corresponds to 1 kilometer. The counter increments every 100 milliseconds. The subsequent clock pulse of the mod-10 counter drives a 4-bit PIPO (Parallel In Parallel Out) register. Initially, within 5 nanoseconds, the stored value in the PIPO is cleared by enabling the clear pin. For the remainder of the cycle, the clear pin is disabled. The clock input for the PIPO is sourced from the output of the mod-10 counter. Hence, whenever the mod-10 counter overflows, the PIPO captures the input value. This value remains stored in the PIPO until the clock signal changes. Subsequently, the value from the PIPO is directed to one of the inputs of a full adder (A0, A1, A2, A3). In the configuration, all four pins of the adder are grounded as they are unnecessary. The other input (B0, B1, B2, B3) is set to 3 because 30 is added for each kilometer, but the units place always remains 0. The output of the adder is fed back to the PIPO to store its value for the next overflow or kilometer.

RESULTS AND DISCUSSIONS

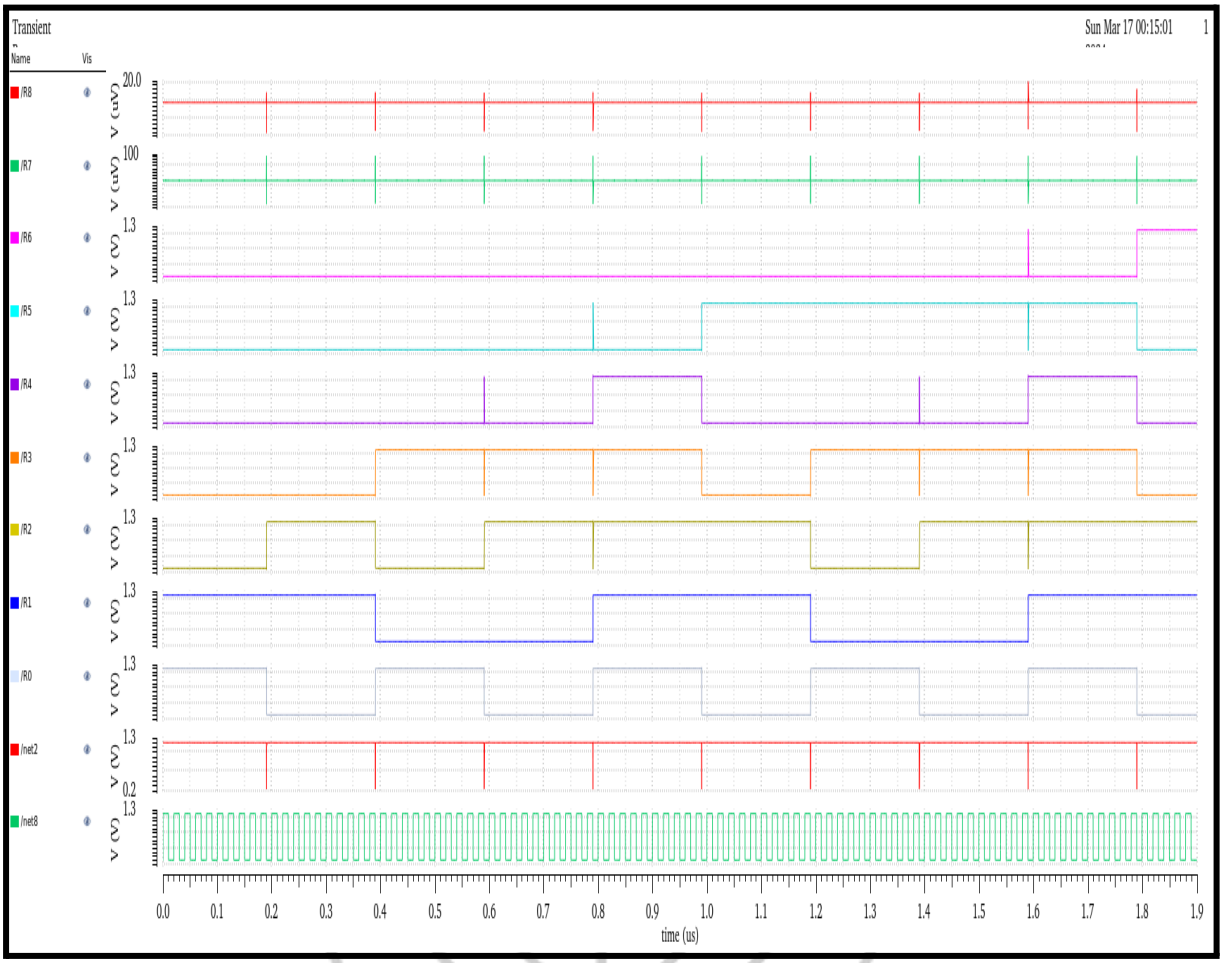

Fig 9: Final output

Initially, the input clock is supplied to the mod-10 counter. The input clock parameters are as follows: V1=0V, V2=1.2V, period=20ns, and pulse width=10ns. Every 200ns, the mod-10 counter overflows, and the subsequent clock signal, which falls and then rises, acts as the clock for the PIPO. Consequently, for every 200ns, the full adder adds 3, indicating that the taxi has traveled 1 kilometer. This progression is observable in the output graph, derived from the outputs of the full adder from R0 to R8.

LEARNING OUTCOMES WITH RESPECT TO COURSE OUTCOMES

1. Practical Application of Components:

Participants gained practical experience in integrating essential electronic components, such as Flip-Flops, Latches, counters, and logical gates, into the taxi fare meter system. This hands-on exposure deepened their understanding of how these components function and their relevance in real-world control systems.

2. System Integration and Interfacing:

The project enhanced participants' ability to integrate various components into a unified taxi fare meter system. Understanding how different elements, like the mod-10 counter and PIPO register, interface with each other provided insights into system-level design considerations.

3. Signal Processing and Feedback Mechanisms:

Participants learned about signal processing within the context of the taxi fare meter. The comparison of signals from the mod-10 counter with predefined thresholds for kilometer detection highlighted the importance of feedback mechanisms in ensuring accurate fare calculation.

4. Simulation and Analysis Techniques:

The use of simulation tools like Cadence enabled participants to set up simulations, define parameters, run analyses, and interpret results within the context of the taxi fare meter. This experience equipped them with transferable skills applicable to other electronic design and simulation tools.

5. Troubleshooting and Debugging:

The simulation process offered participants the opportunity to identify and resolve issues within the taxi fare meter circuit design. This skill is essential in real-world scenarios where debugging and refining designs are common engineering tasks.

6. Application of Control Logic:

The implementation of logical gates to control fare calculation based on distance traveled demonstrated the application of control logic in real-time scenarios. This knowledge is crucial for designing efficient and responsive fare calculation systems for taxis.