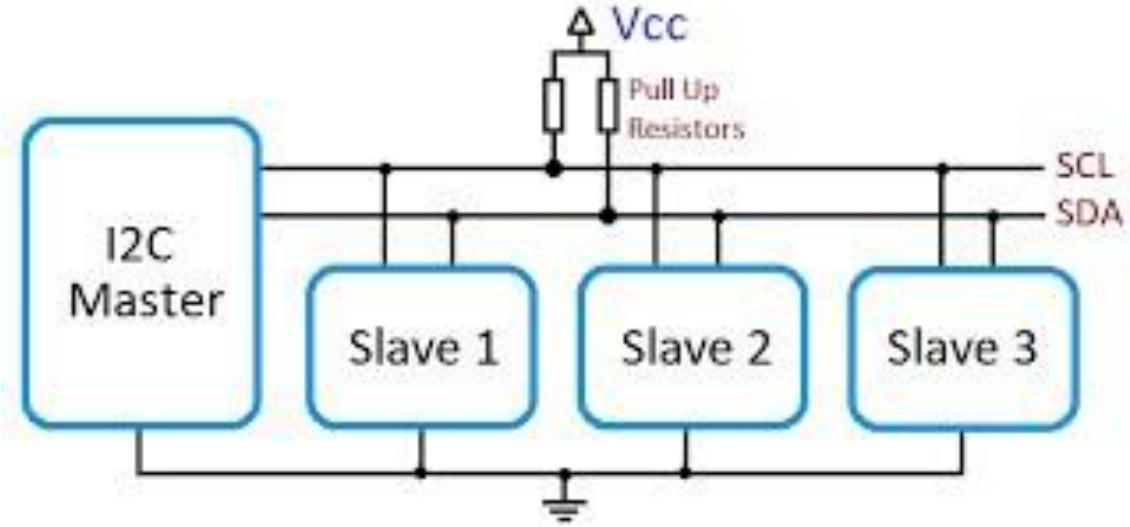
Architecture Overview

What is I2C?

I2C (Inter-Integrated Circuit) is a synchronous, multi-master, multi-slave, packetswitched, single-ended, serial communication bus. Developed by Philips Semiconductors (now NXP Semiconductors), it allows multiple chips to communicate with each other using just two wires: SDA (Serial Data Line) and SCL (Serial Clock Line). I2C is widely used for communication between microcontrollers and peripherals like sensors, displays, and EEPROMs.

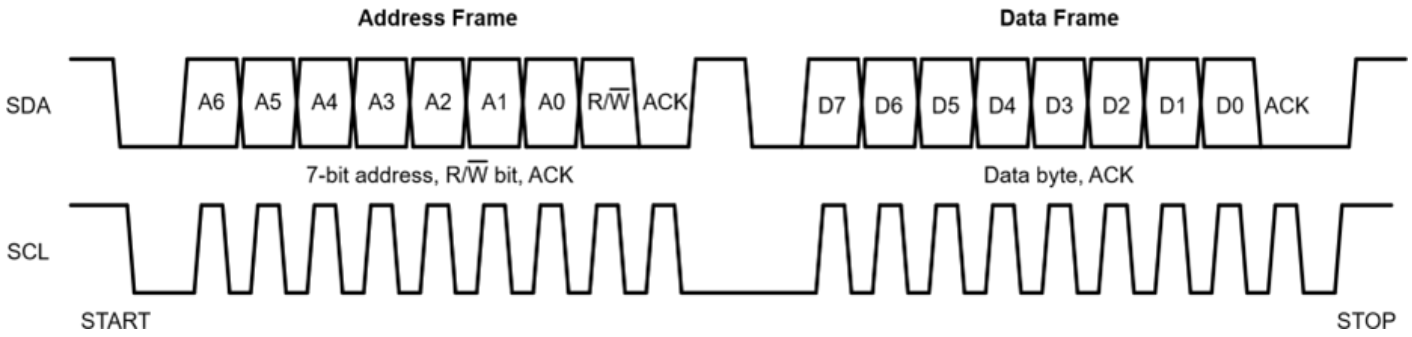
I2C Bus Configuration:

- **Data Validity**: Data on the SDA line must be stable during the HIGH period of the clock. Data line changes are permitted only when the clock is LOW.
- **Start and Stop Conditions**:
  - **Start Condition**: A HIGH to LOW transition on SDA while SCL is HIGH.
  - **Stop Condition**: A LOW to HIGH transition on SDA while SCL is HIGH.
- **Byte Format**: Each byte (8 bits) transferred on the I2C bus is followed by an acknowledgment bit (ACK/NACK).
- **Acknowledge (ACK) and Not Acknowledge (NACK)**:
  - **ACK**: The receiver pulls the SDA line LOW to indicate successful reception of a byte.
  - **NACK**: The receiver leaves the SDA line HIGH to indicate unsuccessful reception or the end of communication.



I2C Protocol Format Overview:

The I2C protocol format specifies how data is transmitted and received over the I2C bus. It consists of several stages, including the start condition, address frame, data frames, and stop condition. Below is a detailed breakdown of each component of the I2C protocol format:



Start Condition:

- The start condition signifies the beginning of communication on the I2C bus.
- It is indicated by a transition from HIGH to LOW on the SDA line while the SCL line is HIGH.
- This alerts all connected devices that a communication session is about to start.

Address Frame:

- Following the start condition, the master sends an address frame to identify the intended slave device.
- The address frame consists of a 7-bit or 10-bit address followed by a read/write bit.
    - **7-bit addressing**: Most common, where the first 7 bits indicate the slave address, and the 8th bit indicates the read (1) or write (0) operation.
    - **10-bit addressing**: Used for systems with more devices, where the first 10 bits represent the address.

Acknowledge (ACK) / Not Acknowledge (NACK):

- After each byte (address or data) is transmitted, the receiving device must acknowledge receipt.
- **ACK**: The receiver pulls the SDA line LOW during the acknowledgment clock pulse.
- **NACK**: The receiver leaves the SDA line HIGH during the acknowledgment clock pulse, signaling the end of communication or that it cannot receive further data.

Data Frames:

- Data frames are 8-bit bytes transmitted after the address frame.
- Each data byte is followed by an ACK/NACK bit from the receiver.
- Data can be transmitted by either the master or the slave, depending on the read/write bit in the address frame.

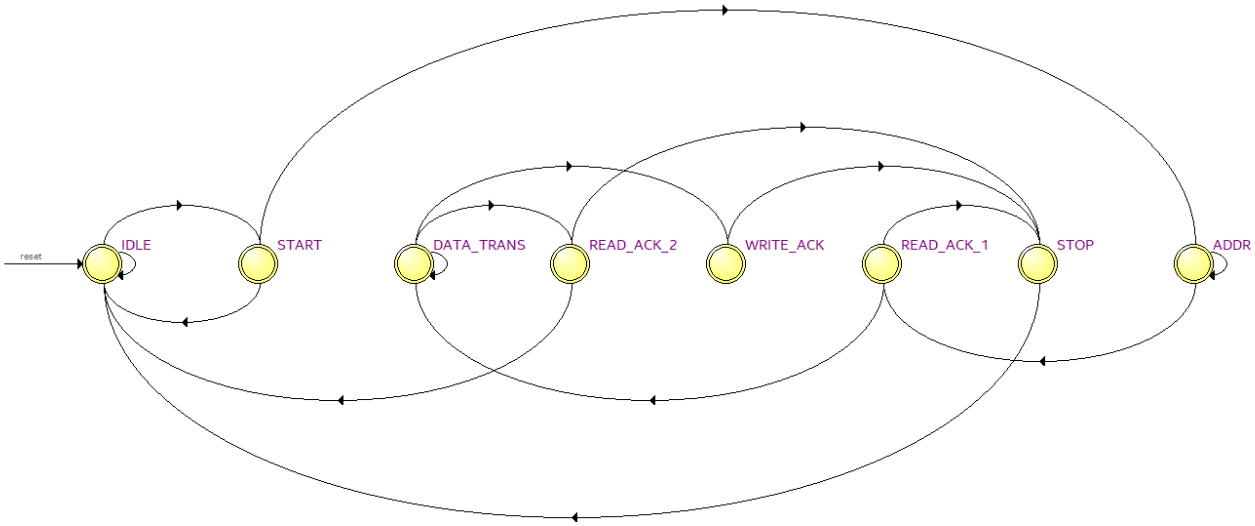
Stop Condition:

- The stop condition signifies the end of communication on the I2C bus.
- It is indicated by a transition from LOW to HIGH on the SDA line while the SCL line is HIGH.
- This releases the bus and allows other devices to initiate communication.

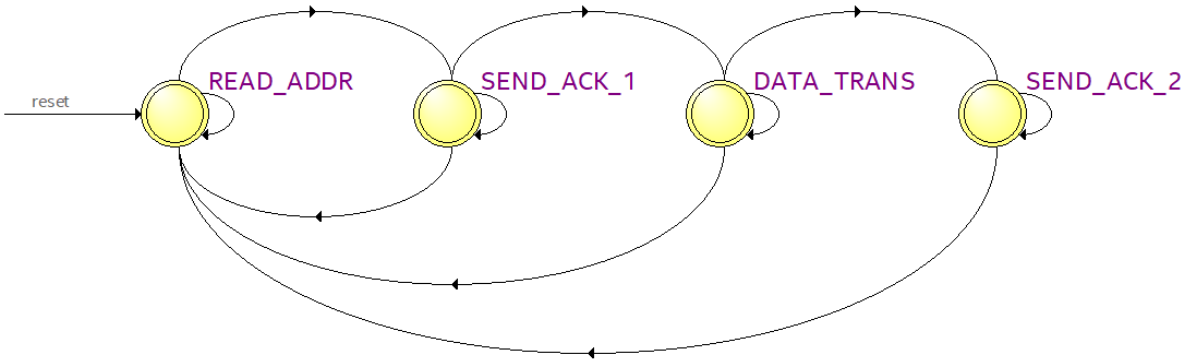
Repeated Start Condition:

- Sometimes, the master needs to keep control of the bus for another transfer without releasing it.
- The repeated start condition is similar to the start condition but occurs without a preceding stop condition.
- It allows the master to change direction (from write to read or vice versa) or address another slave device without releasing the bus.

I2C Master FSM



I2C Slave FSM



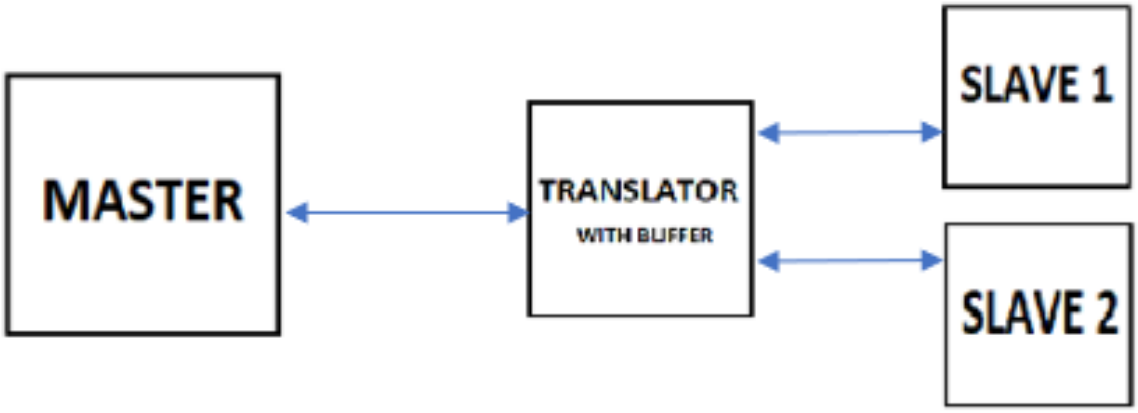
Design Approach

First, I implemented a simple I²C protocol. It uses a Serial Data (SDA) line for data transfer and a Serial Clock (SCL) line to synchronize operations. Devices on the bus can be either a master (initiates communication) or a slave (responds to the master).

To implement the I²C Address Translator, I considered how to handle the flow of data between the system master and the target slave device while allowing address substitution. I approached this problem with two possible methods:

Method 1: Buffer-Based Translation (Used in rtl code)

In this method, the translator first captures incoming data into a buffer (shift register or FIFO). The address byte is modified inside the buffer and then forwarded to the target slave. Data coming back from the slave is also passed through the buffer before being sent to the master.



Method 2: Transparent Pass-Through (MUX Style)

In this method, the translator acts like a direct path (wire/MUX). The address byte is detected and modified on-the-fly, while the rest of the data is passed transparently between master and slave without buffering.

Design Challenged Faced

1. **Bidirectional Data Handling :** Bus contention on the SDA line if the tri-state buffer is not released or enabled at the precise moment required by the I^2C protocol.

2. **Acknowledgment (ACK) :** The challenge of managing a dual role, where you must send an ACK to the bus master while also correctly receiving and relaying the ACK/NACK from the target slave device.