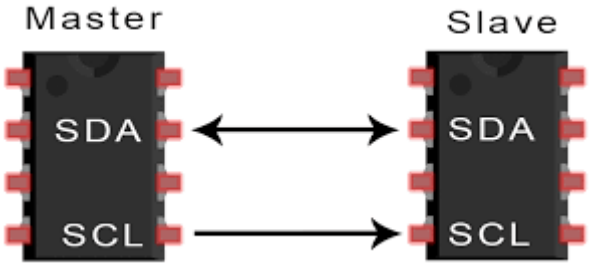
CHAPTER 1: INTRODUCTION TO I²C PROTOCOL

1.1 What is I²C?

I²C (Inter-Integrated Circuit, pronounced "I-squared-C" or "I-two-C") is a **two-wire serial communication protocol** developed by Philips (now NXP Semiconductors) in **1982**.
It allows multiple chips (integrated circuits) to communicate with each other **using just two lines**:



- **SDA (Serial Data line)** – for sending and receiving data
- **SCL (Serial Clock line)** – for timing and synchronization

It is especially useful when **multiple devices** (sensors, displays, EEPROMs, etc.) need to be connected to a single processor without using many pins.

Think of it like a **shared conversation line** where only two "wires" are used for everyone to talk and listen but only one speaks at a time, and the rest listen.

1.2 Why is I²C Used?

- **Saves wiring** – Only 2 wires for many devices
- **Supports multiple devices** – Each device has a unique address
- **Easy to implement in hardware** – widely supported in microcontrollers and FPGAs
- **Flexible speed options** – from slow sensors to fast peripherals

It's like having **one bus route** that can carry different passengers (devices) without adding extra roads (wires).

1.3 History and Evolution

Year	Milestone in I²C Development
1982	I²C protocol introduced by Philips (NXP) at 100 kbit/s (Standard-mode).
1992	Fast-mode (400 kbit/s) introduced for faster devices.
1998	High-speed mode (3.4 Mbit/s) launched for demanding applications.
2007	Ultra-fast mode (5 Mbit/s, write-only) added for displays/LED drivers.
Present	I²C is standard in almost all microcontrollers, FPGAs, and SoCs, often combined with other protocols like SPI, UART, and CAN.

1.4 Applications of I²C

I²C is widely used in:

- **Consumer electronics** – TVs, smartphones, digital cameras

- **Sensors** – temperature, humidity, pressure, accelerometers
- **Memory devices** – EEPROMs, FRAM
- **Displays** – OLED, LCD modules
- **VLSI design** – integrating I²C controllers as IP blocks in SoCs

Example: In a smartphone, I²C connects the **CPU** to the **accelerometer** and **touch controller** so the phone can detect orientation and touch input.

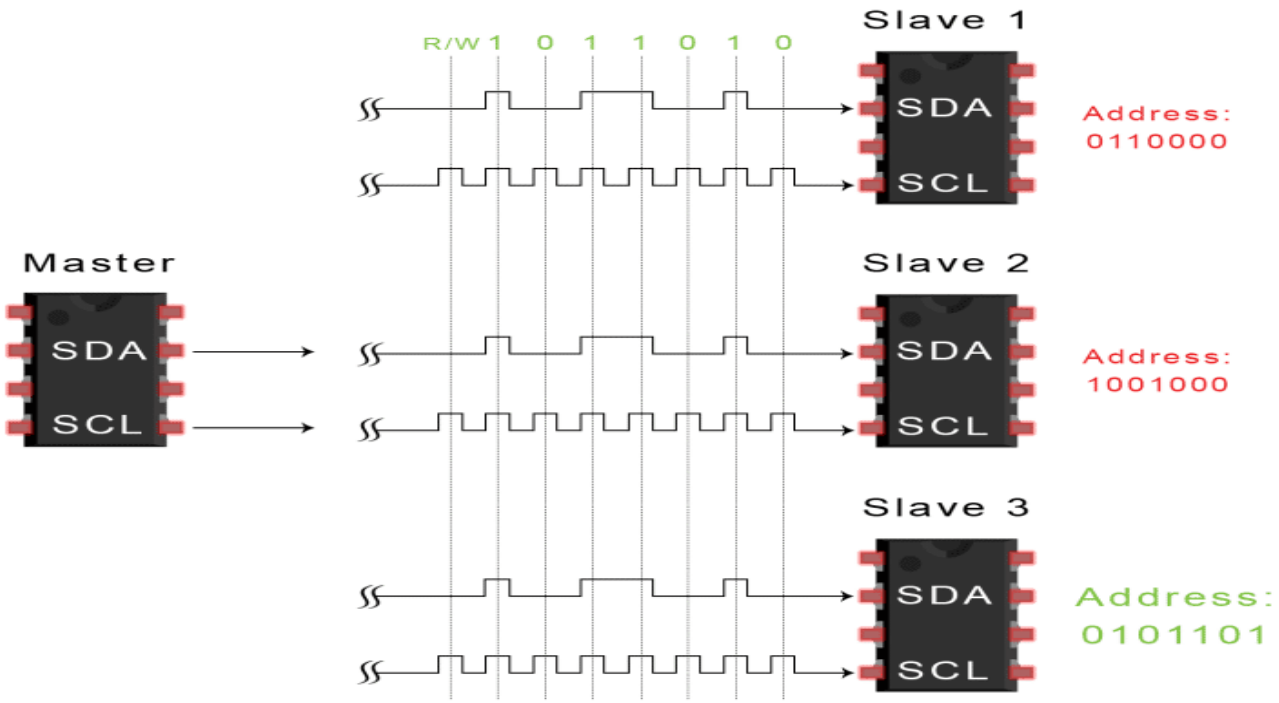
1.5 Role in VLSI

In **VLSI (Very Large-Scale Integration)**, I²C is often implemented as:

- **Controller IP** – placed inside SoC to communicate with external chips
- **Peripheral IP** – sensor or memory chip that connects to the SoC
- Used during **design and verification** in simulation tools like **System Verilog/UVM** to test data transfer correctness, timing, and error handling.

CHAPTER 2: TECHNICAL OVERVIEW OF I²C

2.1 Basic Working Principle



I²C uses **two wires** to transfer data between devices:

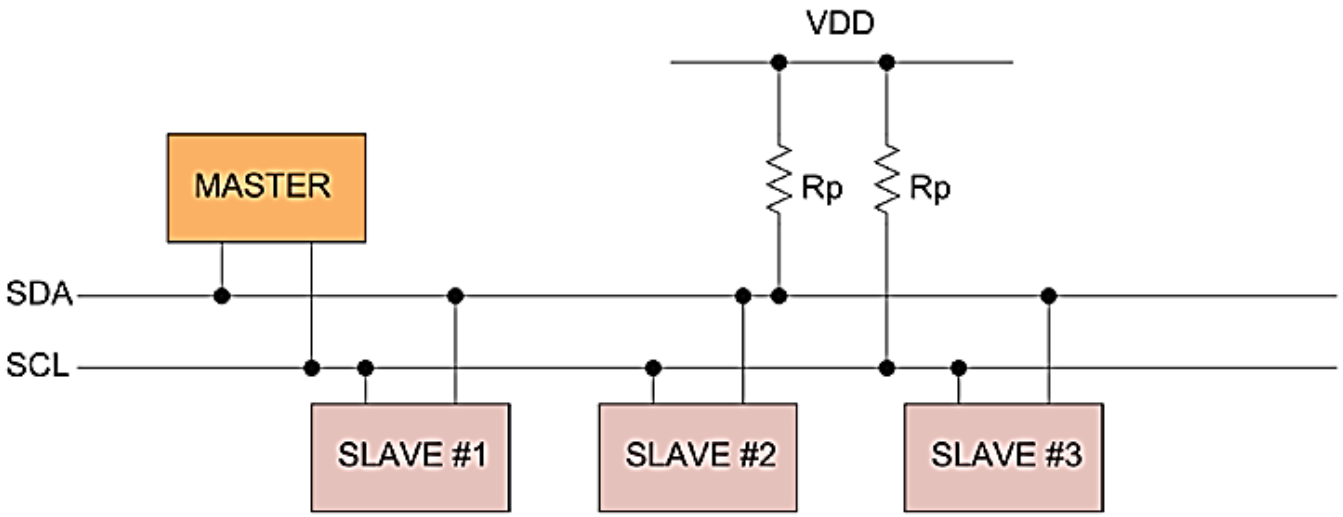
- **SDA (Serial Data line)** – carries the actual bits of data.
- **SCL (Serial Clock line)** – tells when to read/write each bit.

One device act as the **Master** (controls the clock and communication), and others act as **Slaves** (respond when addressed).
Multiple masters can exist, but only one controls the bus at a time (handled by arbitration).

2.2 Two-Wire Architecture

Here is the basic setup:

markdown



Multiple Slaves

- **Pull-up resistors** are needed on both SDA and SCL lines (because I²C uses **open-drain** outputs).
- Any device can pull the line LOW, but releasing it lets it go HIGH (due to pull-ups).

Analogy: Imagine a rope where anyone can pull it down (LOW), but when nobody pulls, it goes back up (HIGH).

2.3 Master–Slave Communication Model

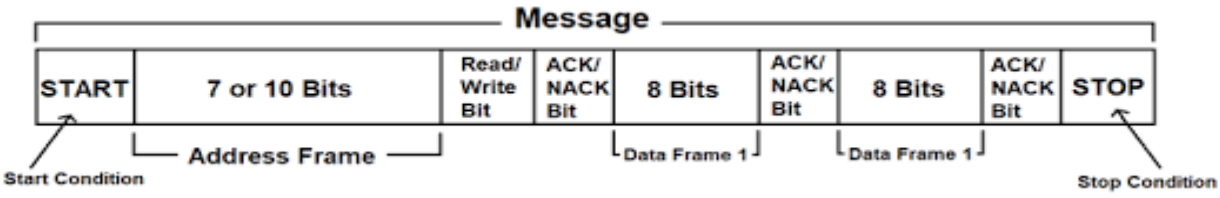
1. **Master initiates communication** by sending a START condition.
2. **Master sends address** of the target slave device.
3. **Slave responds** with ACK (acknowledge).
4. Data is exchanged (read/write) between master and slave.
5. Communication ends with a STOP condition.

2.4 I²C Data Frame Structure

Each transfer is made up of **8-bit data chunks** followed by an **ACK/NACK bit**.

Frame breakdown:

- **Start condition (S)**
- **7-bit or 10-bit address** + R/W bit (0 = write, 1 = read)
- **ACK bit** from slave
- **Data byte(s)** + ACK/NACK
- **Stop condition (P)**

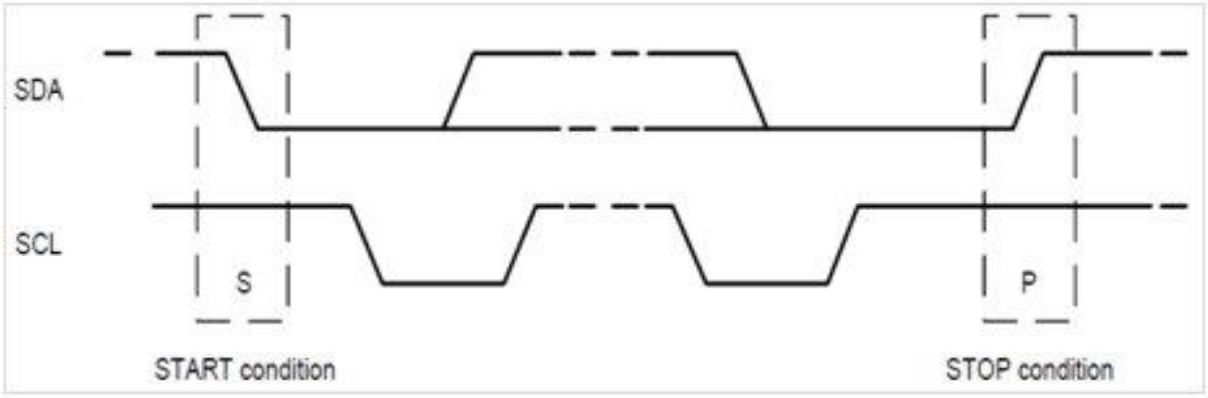


Example:
If the master writes 1010101 (7-bit address) with write = 0, it sends:
1010101 0 → slave ACK → data bytes.

2.5 Timing: Start and Stop Conditions

- **Start Condition (S)**: SDA goes LOW while SCL is HIGH → Signals a new transfer.
- **Stop Condition (P)**: SDA goes HIGH while SCL is HIGH → Signals the end of transfer.

Diagram:

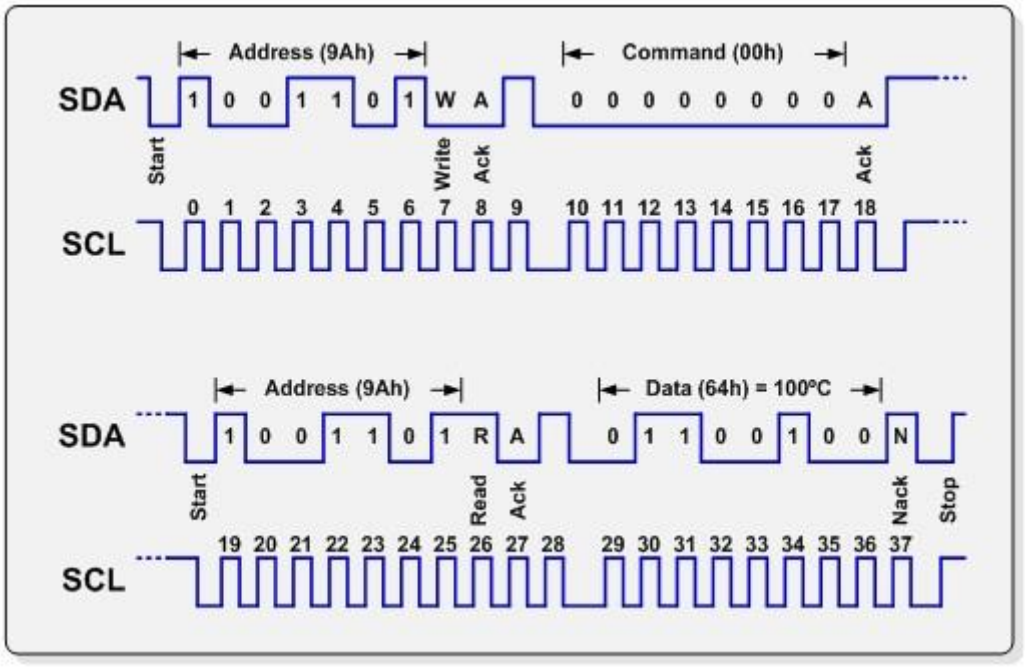


S: SDA ↓ while SCL ↑

P: SDA ↑ while SCL ↑

2.6 Acknowledge (ACK) and Not Acknowledge (NACK)

- **ACK (0)** – receiver pulls SDA low after each byte to confirm reception.
- **NACK (1)** – receiver leaves SDA high, signaling "no more data" or "error".



2.7 Multi-Master and Arbitration

If two masters try to use the bus at the same time:

- They compare bits while sending.
- If one sends a HIGH but detects a LOW, it stops (loses arbitration).
  This ensures **no data collision**.

2.8 Summary Table of I²C Signals

Signal	Purpose	Controlled by	Active Level
SDA	Data line	Master/Slave	LOW (open-drain)
SCL	Clock line	Master	LOW (open-drain)
START	Begin transfer	Master	SDA LOW when SCL HIGH
STOP	End transfer	Master	SDA HIGH when SCL HIGH
ACK	Data confirmation	Receiver	SDA LOW

CHAPTER 3: I²C SPECIFICATIONS AND VARIANTS

3.1 Speed Modes of I²C

Over time, I²C evolved to support different **data transfer speeds** to meet various application needs.

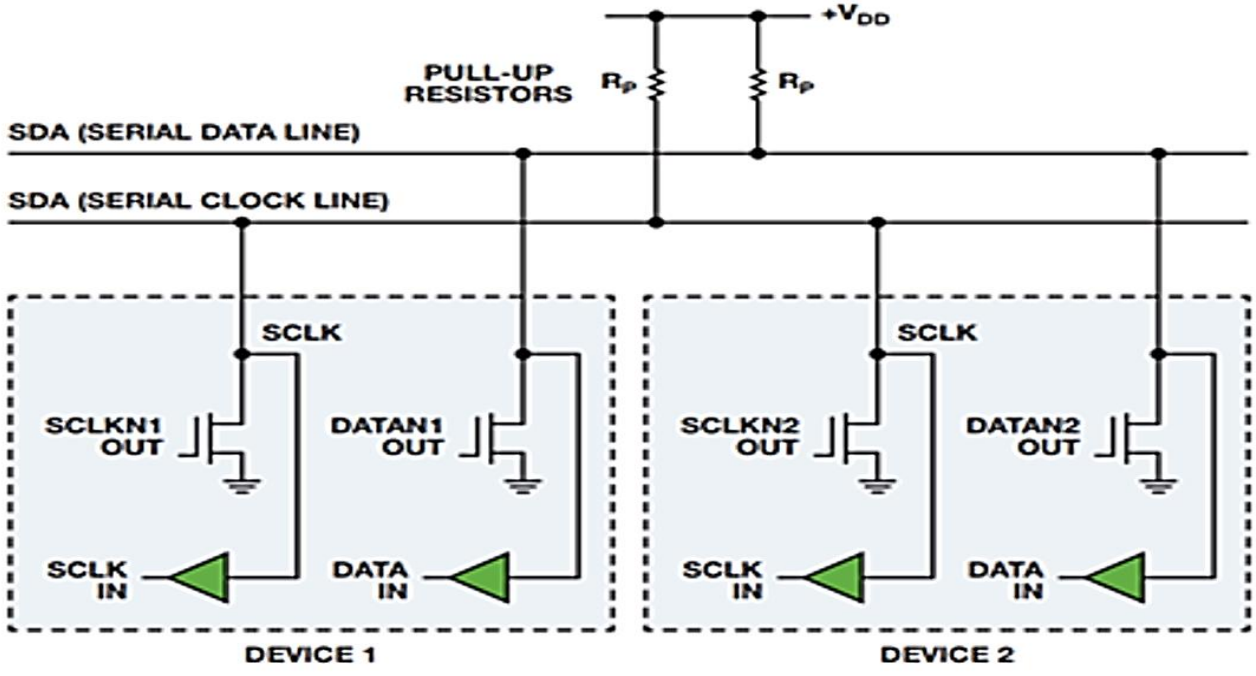
Mode	**Max Speed**	**Typical Use**
Standard-mode (Sm)	100 kbit/s	Simple sensors, EEPROMs
Fast-mode (Fm)	400 kbit/s	Faster sensors, LCD controllers
Fast-mode Plus (Fm+)	1 Mbit/s	High-performance sensors
High-speed mode (Hs-mode)	3.4 Mbit/s	Advanced displays, high-speed peripherals
Ultra-fast mode (UFm)	5 Mbit/s (write-only)	LED drivers, fast DACs

Example:

- A temperature sensor might use **Standard-mode**,
- An OLED display might use **Fast-mode**,
- A high-speed DAC might use **Ultra-fast mode**.

3.2 Electrical Characteristics

I²C is based on **open-drain (or open-collector)** outputs:



- Devices can only **pull the line LOW**.
- Lines are **pulled HIGH** through **pull-up resistors**.
- This avoids conflicts when multiple devices are connected.

Why pull-up resistors are important:
Without them, SDA and SCL would just "float" and not register proper logic levels.

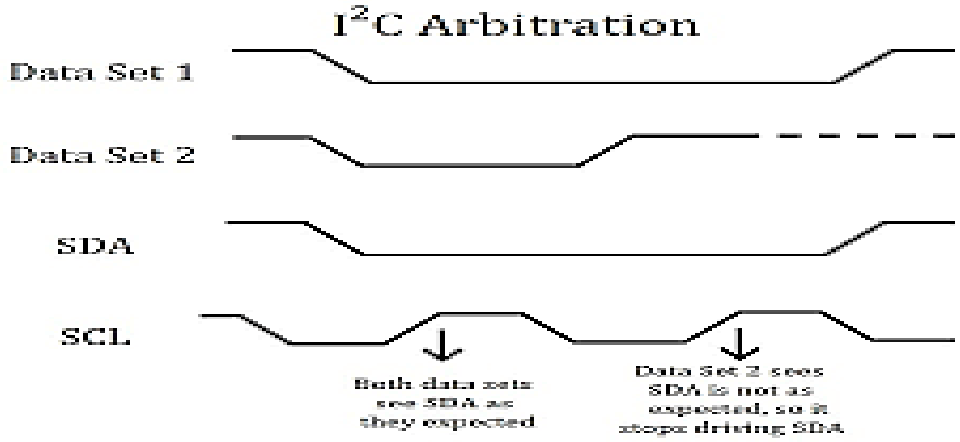
3.3 Addressing Modes

- **7-bit Addressing** – Supports up to 128 devices (most common).
- **10-bit Addressing** – For systems needing more devices.
  Every device has a **unique address**, so the master knows who to talk to.

3.4 Multi-Master Support

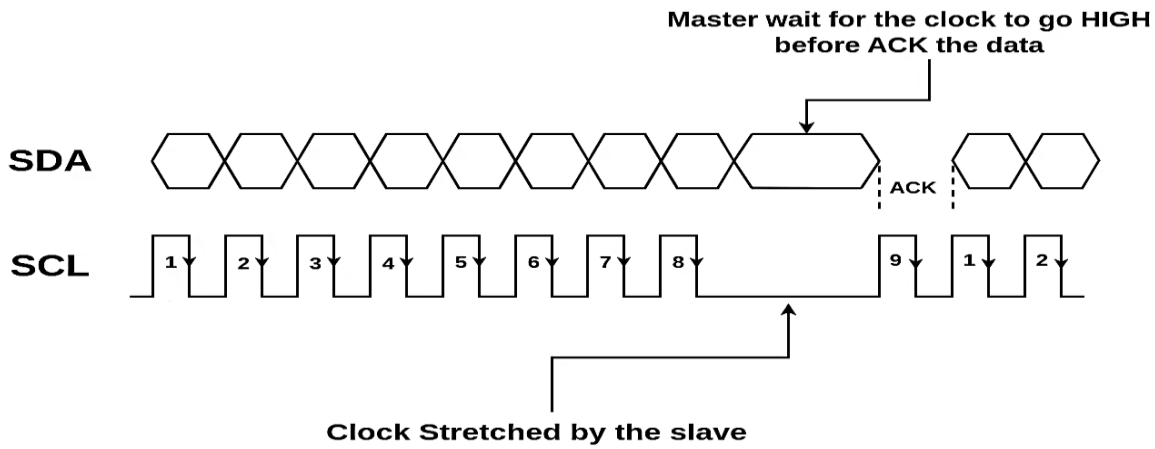
I²C allows **more than one master**, but:

- Only one master can control the clock at a time.
- Arbitration decides who gets control without corrupting data.
- Very useful in advanced SoC designs where multiple controllers exist.



- Clock Stretching During Communication, on a byte level, device may be able to receive data at fast rate but it needs more time to store a received byte or to prepare a next byte to be transmitted. Slaves can then hold the SCL line low which is known as Clock Stretching. During this time, master goes into a wait state.

3.5 I²C Variants

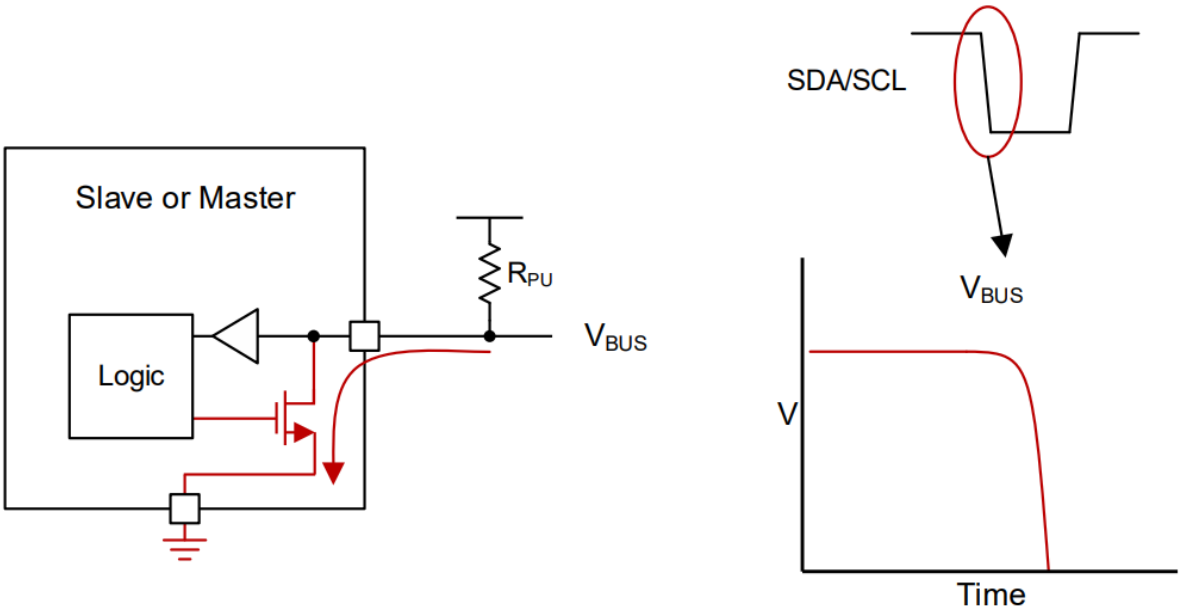


1. **Fast-mode Plus (Fm+)** – Higher speed (1 Mbit/s) with stronger drive capability.

2. **High-speed Mode (Hs-mode)** – Uses special signaling to reach 3.4 Mbit/s.
3. **Ultra-fast Mode (UFm)** – Write-only, 5 Mbit/s, designed for one-way high-speed data.
4. **SMBus (System Management Bus)** – I²C-like protocol with extra rules for power management, mainly in computers.
5. **PMBus (Power Management Bus)** – Based on SMBus, used for digital power control.

3.6 Typical I²C Connection Diagram

- [R] = Pull-up resistors (value depends on speed and bus length).
- **All devices share SDA & SCL**.



3.7 Summary Table of I²C Variants

Variant	**Max Speed**	**Direction**	**Common Use**
Standard-mode	100 kbit/s	R/W	Low-speed sensors
Fast-mode	400 kbit/s	R/W	LCD, EEPROM
Fm+	1 Mbit/s	R/W	Industrial sensors
Hs-mode	3.4 Mbit/s	R/W	High-speed peripherals
UFm	5 Mbit/s	Write	LED drivers
SMBus	100-400 kbit/s	R/W	PC management
PMBus	100-400 kbit/s	R/W	Power management

CHAPTER-4: SUMMARY, TRENDS, AND FUTURE SCOPE

4.1 Summary of Key Points

- **I²C Basics** – A simple two-wire serial communication protocol (SDA & SCL) developed by Philips in 1982 for connecting multiple ICs using minimal wiring.

- **Master–Slave Architecture** – One master controls the bus, while multiple slaves respond when addressed.
- **Data Frames** – Communication happens in address + data bytes, with ACK/NACK for reliability.
- **Speed Modes** – From **Standard-mode (100 kbit/s)** to **Ultra-fast mode (5 Mbit/s)** to suit different applications.
- **VLSI Integration** – I²C is implemented as an IP block in SoCs, designed in RTL, and verified using methodologies like SystemVerilog + UVM.
- **Advantages** – Simple wiring, multi-device support, low cost.
- **Limitations** – Short distance, limited speed compared to SPI or UART.

4.2 Current Trends in I²C Technology

- **Higher-speed adoption** – More devices use Fast-mode Plus (1 Mbit/s) and High-speed mode (3.4 Mbit/s) for better performance.
- **Integration in SoCs** – I²C controllers are now standard in microcontrollers, FPGAs, and custom ASICs.
- **Mixed-protocol designs** – Modern boards combine I²C with SPI, UART, and CAN for hybrid communication systems.
- **Low-power designs** – Optimized I²C controllers for battery-powered devices (wearables, IoT sensors).
- **Automotive & Medical use** – I²C is being used in automotive ECUs and medical devices for configuration and control.

4.3 Future Scope

- **Improved Robustness** – Better error detection and noise immunity for industrial environments.
- **Longer Bus Lengths** – Research into signal conditioning to extend I²C range beyond a few meters.
- **AI-assisted Bus Monitoring** – Intelligent systems to detect and predict bus failures in real-time.
- **Integration with Emerging Technologies** – I²C coexisting with high-speed serial buses in chiplet-based SoC designs.
- **Smart Verification Tools** – Automated protocol verification for faster chip development cycles.

REFERENCE

1. **UM10204 – I²C-bus Specification and User Manual, Rev. 7.0 (2021)**
   NXP's latest official I²C specification with all speed modes and updates.
   https://www.nxp.com/docs/en/user-guide/UM10204.pdf
2. **I²C-bus.org – Specification Overview (Version 6.0)**
   Central resource for I²C protocol info and related standards.
   https://www.i2c-bus.org/specification/
3. **Kernel.org – Introduction to I2C and SMBus**
   Documentation confirming UM10204 Rev.7 as current spec.
   https://www.kernel.org/doc/html/latest/i2c/summary.html
4. **I²C Revision 7 Notes – Terminology Updates**
   Info on changes like "master/slave" to "controller/target."
   https://forum.arduino.cc/t/new-i2c-standard-document-by-nxp-controller-and-target/913956