Digital Logic Design

Communication Protocols

Outline

- Introduction
- Communication Categorization based on Physical, Logical specification
  - Single-Ended, Differential
  - Communication Modes
  - Serial, Parallel
  - Synchronous, Asynchronous
  - Little, Big Endian
- Example interfaces:
  - UART
  - SPI
  - I2C
  - USB
- Error Detection and Correction

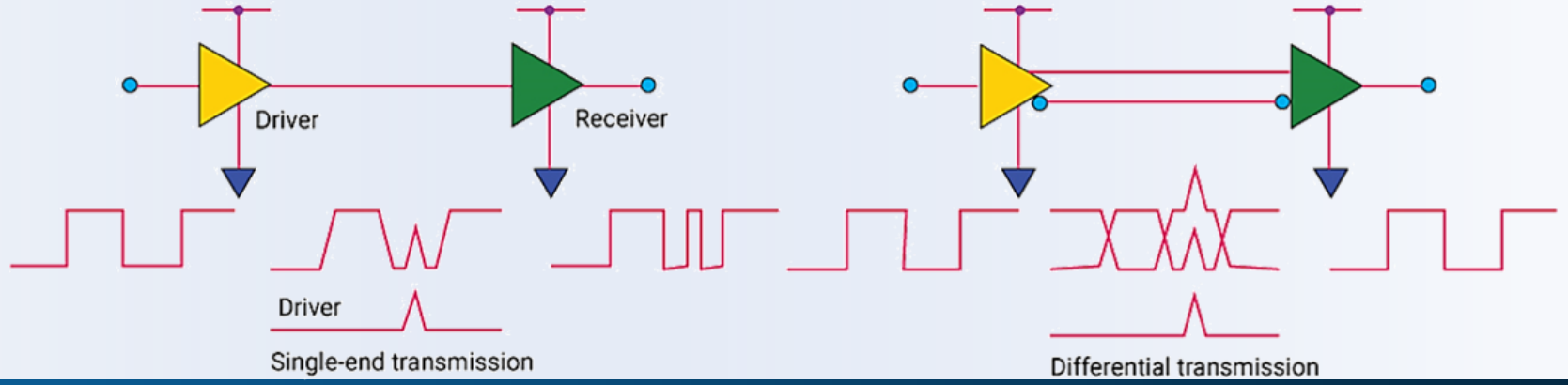
Introduction

- Communication interfaces are used to connect two or more digital components or systems together

- Standardized Interfaces are typically specified using four components: i.e.
  - Electrical: deals with voltages, line capacitance and other electrical characteristics
  - Mechanical: deals with physical description of connector / plug.
  - Functional: describes the function of each pin or circuit in a particular interface.
  - Procedural: describes how particular circuit in an interface perform an operation, e.g. how the data is encoded to electrical signals, and then decoded back to its digital representation, as well as how sender and receiver coordinate the transmission

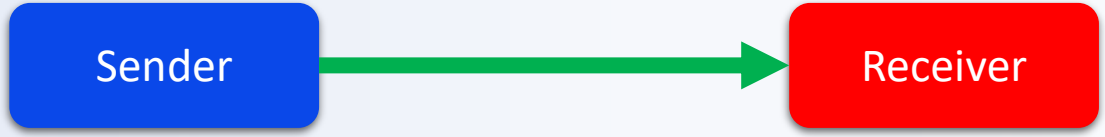
Single-Ended, Differential Interfaces

- Single-Ended Interface
  - Used for interfacing devices/systems at short distance and noise-free environment.
  - Signals are reference to ground, therefore noise may affect.
  - Example: SPI, I2C
- Differential Interface
  - Each signal and its inverse represent a single data line; are closely coupled physically.
  - Immune to noise.
  - Signals are referenced to its respective inverse rather than ground. Noise effects both the signal and its inverse, but the difference remains the same at receiver end.
  - Example: USB, Ethernet, CAN, LVDS, HDMI

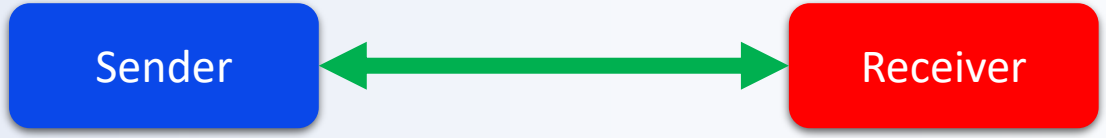


Communication Modes

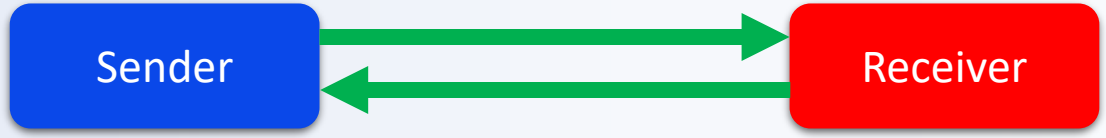
- Simplex
  - One-way communication
  - There is only a fixed sender and a fixed receiver, they cannot change roles
  - Radio and TV are the examples of simplex transmissions



- Half Duplex
  - Bidirectional communication
  - Both sender and receiver can send, but not at the same time, i.e. if a sender transmits, the receiver can accept but cannot send
  - Early telephone modems were half-duplex

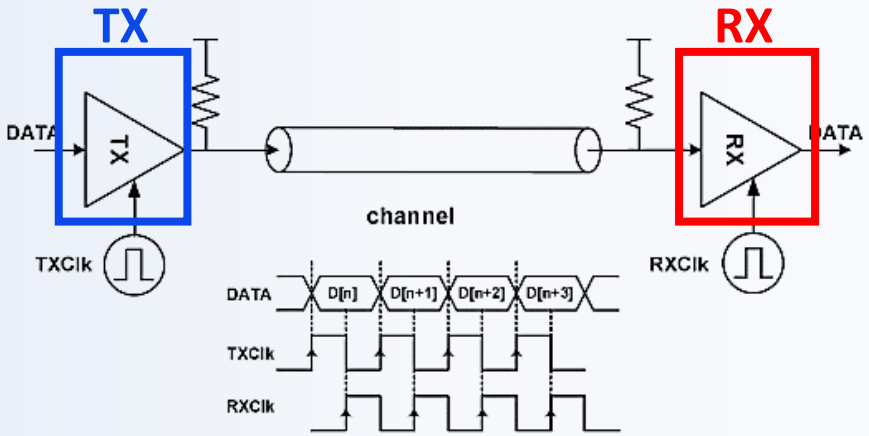


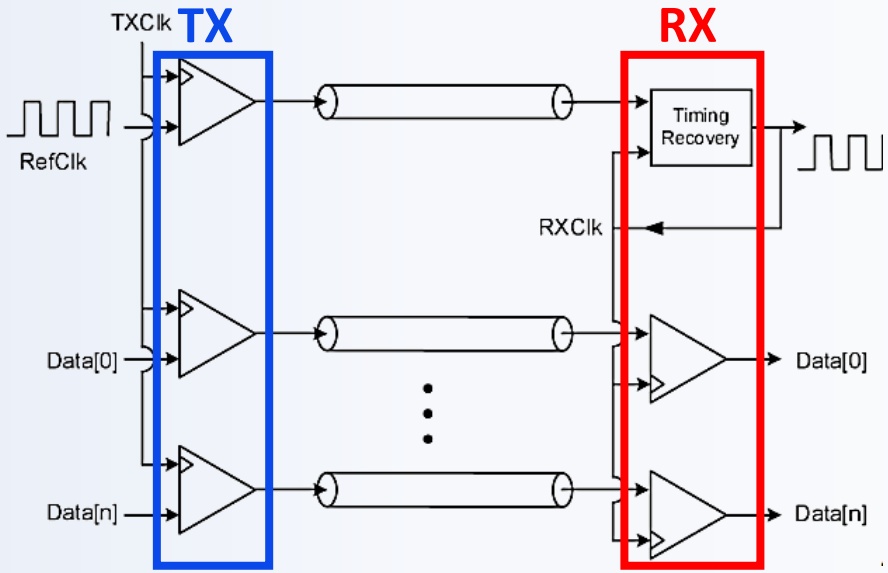
- Duplex
  - Bidirectional communication
  - Both sender and receiver can transmit and receive at the same time



Serial, Parallel – Communication

- Serial
  - Sending bits one after another, instead of all at the same time
  - Uses fewer pins and associated circuitry
  - Slower as compared to parallel communication
  - Examples: SPI, I2C, UART, USB, HDMI, Ethernet, CAN
- Parallel
  - Transmits and receives bits in parallel.
  - Uses as many pins, wires (and associated circuit) as the word size that is transmitted/received.
  - Typically used in on-chip interfaces





Synchronous, Asynchronous – Communication

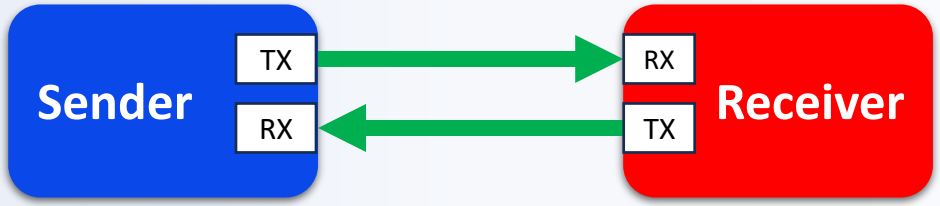
- Synchronous
  - Common clock line between devices to synchronize data transfer.
  - Example: SPI, I2C

- Asynchronous
  - No common clock between devices.
  - Data identified by additional bits at the beginning and end of each data frame.
  - Utilized less resource compared to synchronous interfaces but is comparatively slower.
  - Example: RS-232, RS-485

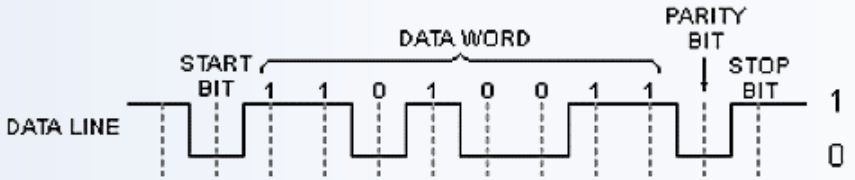
Universal Asynchronous Receiver-Transmitter

- Universal Asynchronous Receiver-Transmitter (UART) allows for asynchronous serial communication.
  - Asynchronous communication for sender or receiver to initiate communication at any time, not synchronized by a common clock.
  - Serial communication sends 1 bit at a time serially over the data line.
  - Full-duplex communication allows for sender and receiver to transmit at the same time.

Block diagram of UART interface

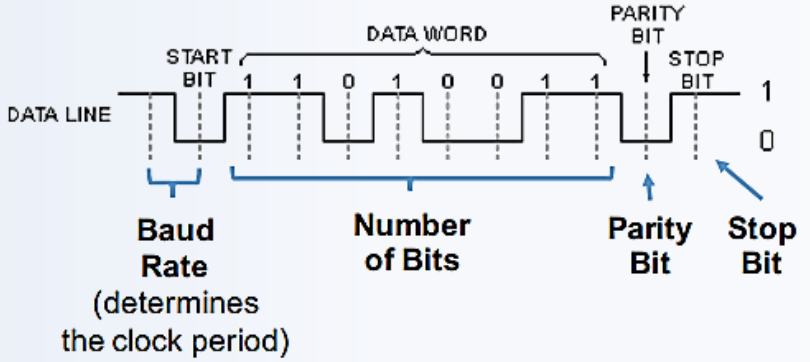


Example Waveform of UART interface



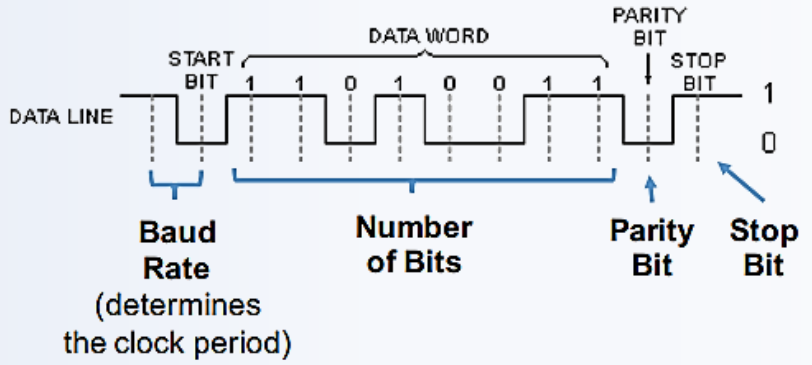
UART – Parameters

- In UART protocol the sender and receiver have to agree on the communication parameters: baud rate, number of bits, parity bit (error checking), data framing.
  - **Baud Rate** specifies how many symbols are transmitted per second.
    - If each symbol is either 1 or 0, then baud rate is the same as bit rate of the communication, i.e. 1 baud = 1 bit/s
    - Typical baud rates: 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1000000, 1500000
  - **Number of Bits** specifies how many bits are transmitted at a time.
    - Typically 8 bits, i.e. 1 byte is sent at a time.
  - **Parity Bit** can be specified for basic error checking.
  - **Data Framing** necessities specification of Start and Stop Bits to detect when the data arrives and when data ends..



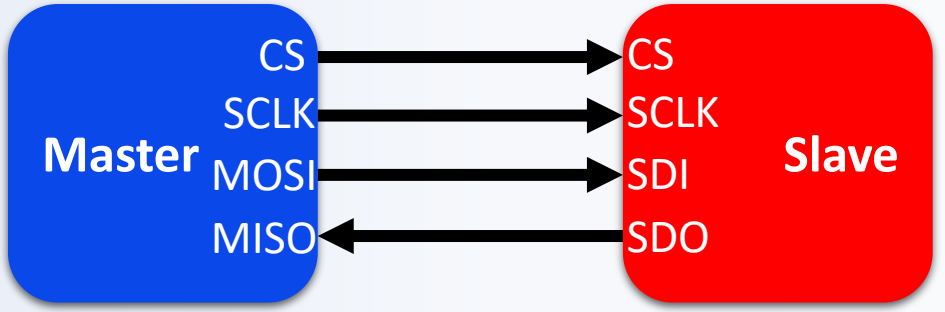
UART – Transmission Operation

1. Sender receives data in parallel from data bus.
2. Sender adds start, parity and stop bits.
3. The entire packet is sent starting from start bit to stop bit from Sender UART to the Receiver UART. The receiver UART samples the data at the preconfigured baud rate.
4. The Receiver UART discards the start, parity and stop bits from the data frame.
5. The Receiver UART coverts the serial data back into parallel and transfers it to the data bus on the receiving end.



Serial Peripheral Interface (SPI)

- One of the most widely used interfaces between microcontroller and peripheral ICs.
- Synchronous, Full-duplex
- Sender and receiver uses shift registers to send and receive the data.
- SPI interface has a **master** device that generates clock for synchronizing the data transmission/reception and can have multiple **slave** devices.
- Slave devices can be connected either in parallel to a single data bus or in daisy-chain method.
- 4-wire SPI has four signals: CLK, CS, MOSI, MISO

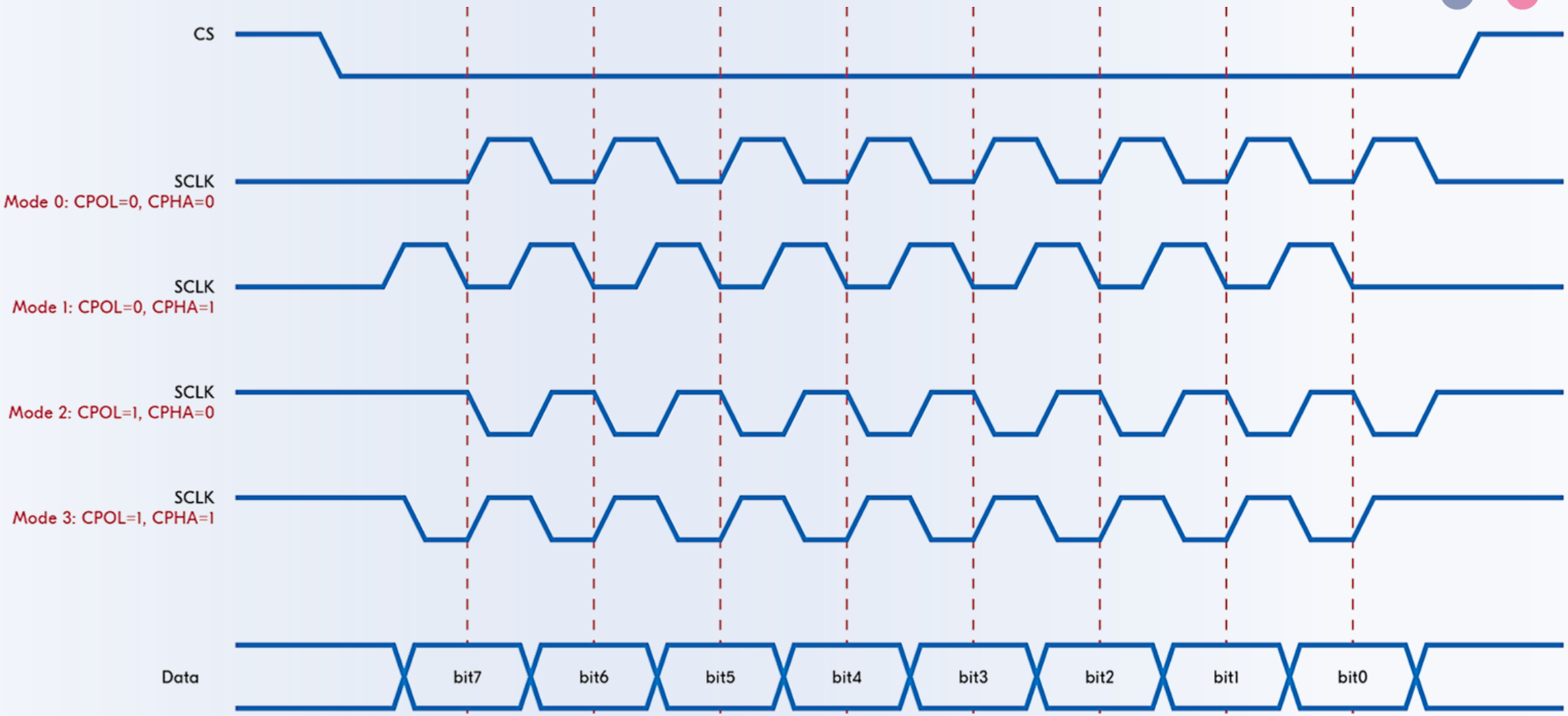


SPI – Modes

- Clock polarity and phase defines the mode of SPI; selected by master

SPI Mode	CPOL	CPHA	Clock Polarity	Clock Phase used to sample and/or Shift the data
0	0	0	Logic Low	Data sampled on rising edge and shifted out on the falling edge
1	0	1	Logic Low	Data sampled on falling edge and shifted out on the rising edge
2	1	0	Logic High	Data sampled on falling edge and shifted out on the rising edge
3	1	1	Logic High	Data sampled on rising edge and shifted out on the falling edge

SPI – Modes (continued)



SPI – Operation

- Chip Select (CS) / Slave Select (SS) signal is used by Master to identify the peripheral device that it wants to talk to.
- Clock synchronizes the data transfer; One bit is transferred in each clock cycle.
- Master sends the data to the peripheral bit by bit, each clock cycle on MOSI line.
- Slave receives according to the SPI mode.
- Slave can send the data to the master through the MISO line serially at the same clock rate.

I2C Interface

- I2C (Inter-Integrated Circuit), also written I2C, is a synchronous serial communication protocol mainly for use between integrated circuits.
- Unlike UART protocol, there is a Controller that initiates all the communication and a Target that only responds to the controller.
  - Previously, and still common, are the terms Master (M) and Slave (S) to represent the two components.
- Like Serial protocol it uses two signals or wires, but in different ways.
  - SDA is the data line.
  - SCL is the clock line.
  - The receiver is simpler as it does not have to generate the clock, only use the SCL line.

I2C – SDA, SCL Lines

- Two lines, or ports, are connected between the target and controller.
  - There is no dedicated transmit or receive line, whoever pulls SDA low while SCL stays high gets to transmit.
  - This requires tri-state logic: logic 1, logic 0, or high-impedance Z.
  - “Outputting” high-impedance basically means there is no connection.
  - Use pull-up resistors to keep SDA and SCL high when everybody is no transmitting.

I2C – SDA, SCL Operation

- In I2C a transaction is initiated by the controller, but a target can respond to a controller's transmission with its own transmission.
- Data is transmitted by:
  - Data transfer is initiated with a start condition (S) by pulling SDA low while SCL stays high.
  - Sender pulls SCL low, while setting SDA to the first bit (low or high).
  - Sender raises SCL, receiver samples the SDA when SCL rises.
  - Repeat for all bits to transfer.
  - After last bit, pull SCL low and SDA low.
  - Finally a stop condition (P) is asserted when SCL rises followed by SDA rising.

Universal Serial Bus (USB)

- Differential, Asynchronous interface.
- Uses a pair of data lines at physical layer, along with 5 V VBUS and GND connections.
- Can communicate at data rates of up to 10 Gbits/s (USB 3.2)
- Uses a tiered architecture to connect up to 127 devices to a single host.
- A typical USB interface implements four basic types of data transfer:
  - Control Transfers: Used to configure a device
  - Bulk Data Transfers: Large size data transfer
  - Interrupt Data Transfers: Used for timely and reliable delivery of data
  - Isochronous Data Transfers: Prenegotiated bandwidth, latency

USB Interface – Operation

- Device is detected using pull-up resistors.
- USB Packet Fields
  - Sync: Sync bit pattern to allow receiver clock to synchronize with data.
  - PID: Packet Identifier bits
  - ADDR: Device address field, typically 7-bit long
  - ENDP: Endpoints, specifying the device's endpoint buffer
  - DATA: Data bytes
  - CRC: Cyclic Redundancy Check bits for Error checking
  - EOP: End of Packet
- At outset, token packet identifies the type of communication to take place, including device address, endpoint, etc.
- Followed by data packet and handshake packet to acknowledge the transaction.

Thank You

NCDC
NUST
CHIP
DESIGN
CENTRE