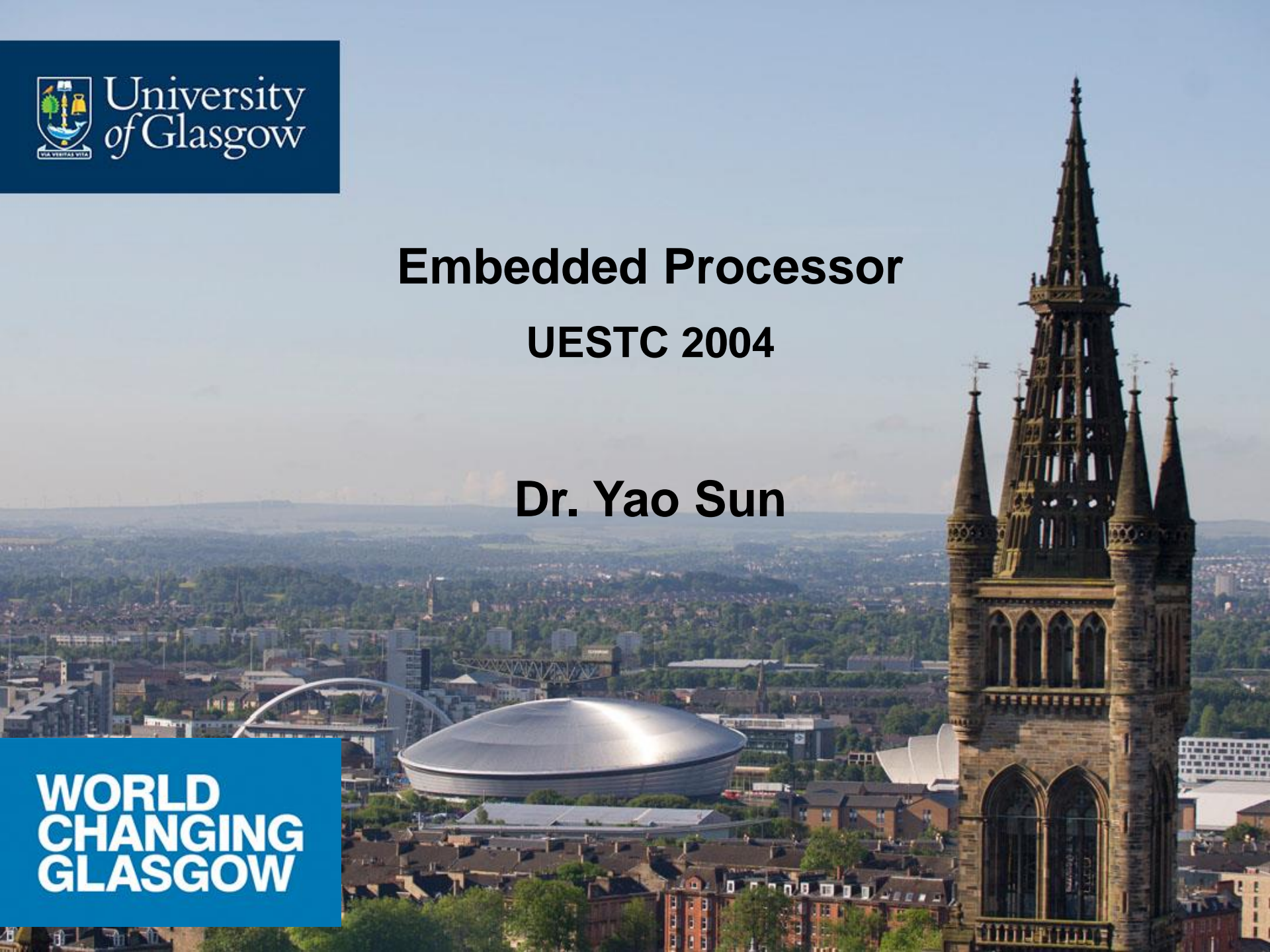


Embedded Processor

UESTC 2004

Dr. Yao Sun

**WORLD
CHANGING
GLASGOW**



BUS

- The basis of BUS
- URAT Protocols
- **I2C Protocols**
- SPI Protocols

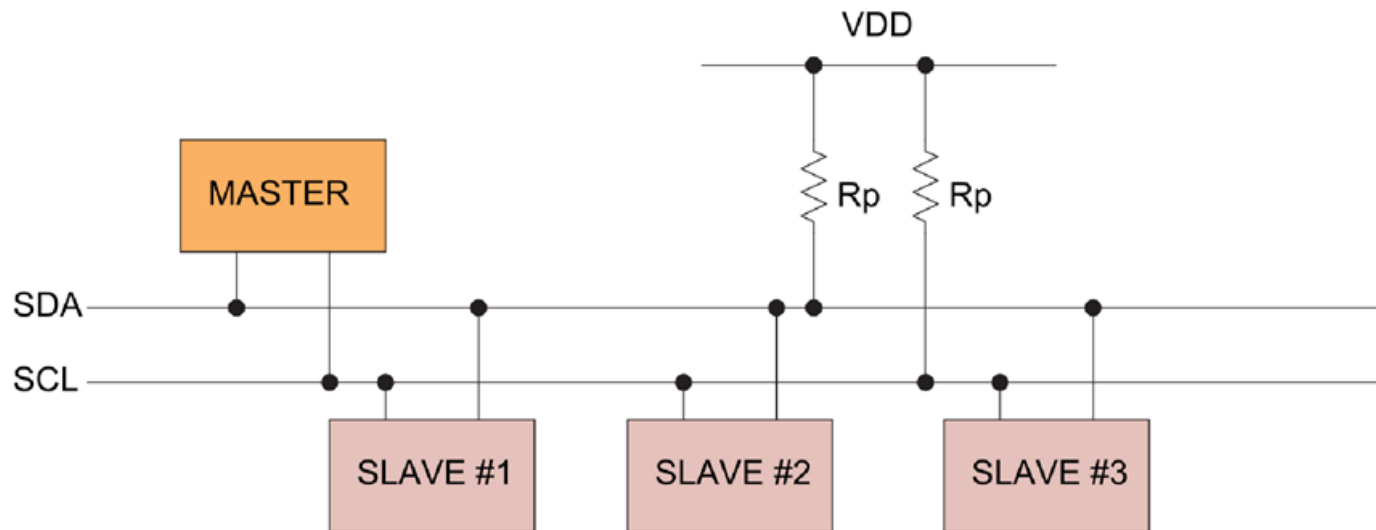
I2C Protocol

➤ What is I2C?

- Inter-integrated-circuit (I2C)
- **Protocol** for **Synchronous** communication in BUS system
- Follows a master/slave hierarchy
- One master, multiple slaves
- Only master can initiate a data transmission
- Widely used for projects that require many different parts (e.g. sensors, pin, expansions and drivers) working together
- The standard data transfer rate is 100kbits/s while the Fast Mode transfer rate is 400kbits/s.

I2C Protocol

➤ Structure of I2C system



- Serial Data Line (SDA): transmit **data, address, and control** signals
- Serial Clock Line (SCL): for synchronization

I2C Protocol

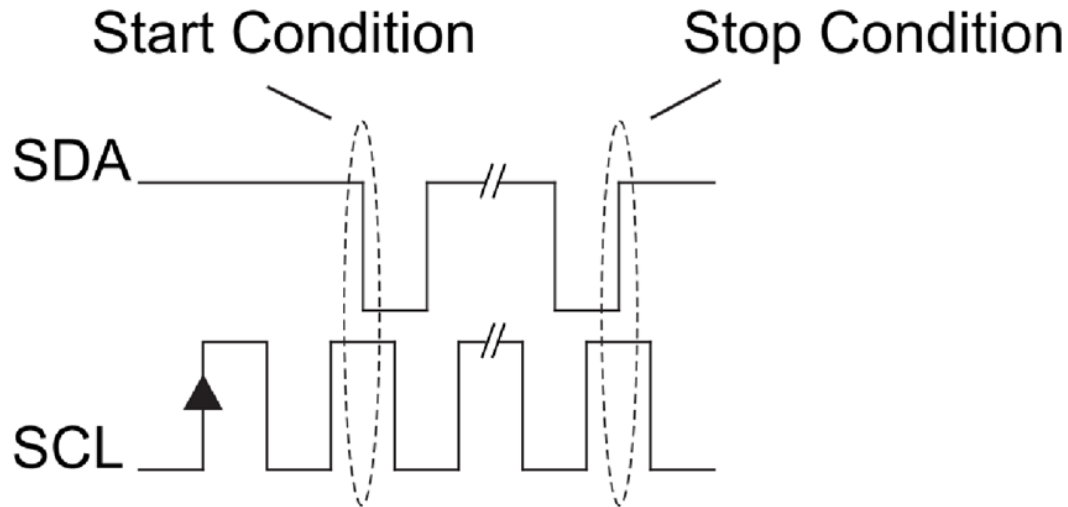
➤ How does I2C work?

- Data length:
 - ✓ I2C data packets are arranged in **8-bit** bytes
- Start condition:
 - ✓ always occurs at the **start of a transmission** and is **initiated by** the **MASTER** device.
 - ✓ wake the **idling SLAVE** devices on the bus
 - ✓ the SDA line transitions **from HIGH** state **to LOW** state, while SCL is **HIGH**.

I2C Protocol

➤ How does I2C work? (cont'd)

- Start condition:



SDA: HIGH → LOW
SCL: HIGH

I2C Protocol

➤ How does I2C work? (cont'd)

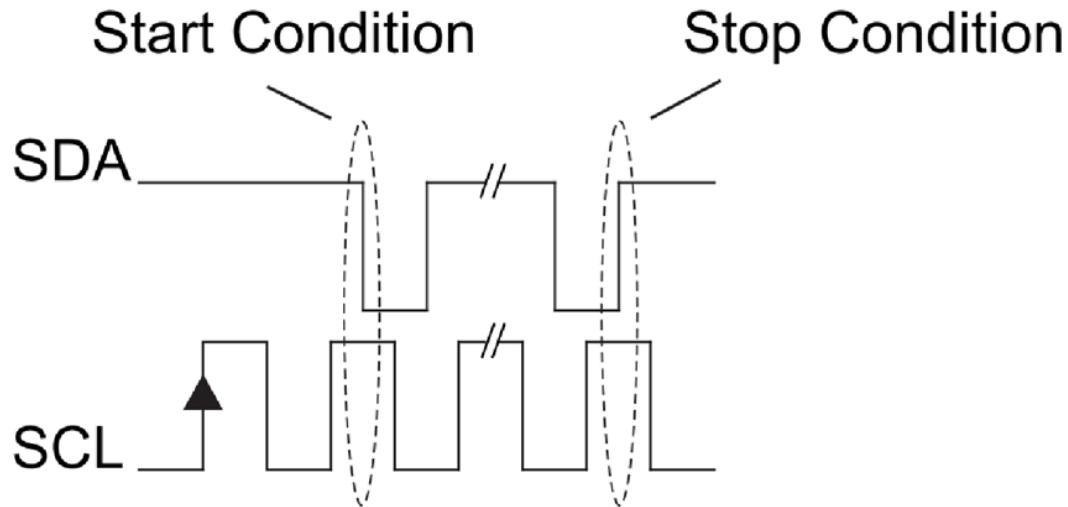
■ Stop condition:

- ✓ occurs at the **end of a data transmission** and is **generated by** the **MASTER** device.
- ✓ tell the **slave devices** that they should go back to an **idle state**
- ✓ the SDA line transitions **from LOW** state **to HIGH** state, while SCL is **HIGH**.

I2C Protocol

➤ How does I2C work? (cont'd)

- Stop condition:



SDA: **LOW** → **HIGH**
SCL: **HIGH**

I2C Protocol

➤ How does I2C work? (cont'd)

■ Address Byte:

- ✓ A slave address is sent in 8-bit byte format
- ✓ the upper 7 bits, constitute the slave address
- ✓ the 8th bit serves as a READ/WRITE# command bit
- ✓ 0: write to a slave; 1: read from a slave

I2C Protocol

➤ How does I2C work? (cont'd)

- Acknowledge and Not Acknowledge Bits (ACK/NACK)
 - ✓ after **every byte** transmission the receiving device sends an ACK or NACK bit.
 - ✓ An ACK is used to denote that a byte (address or data) was transmitted and received **successfully**
 - ✓ A NACK indicates error occurs somewhere
 - ✓ An ACK is generated by the receiver, SDA LOW and SCL HIGH

I2C Protocol

➤ How does I2C work? (cont'd)

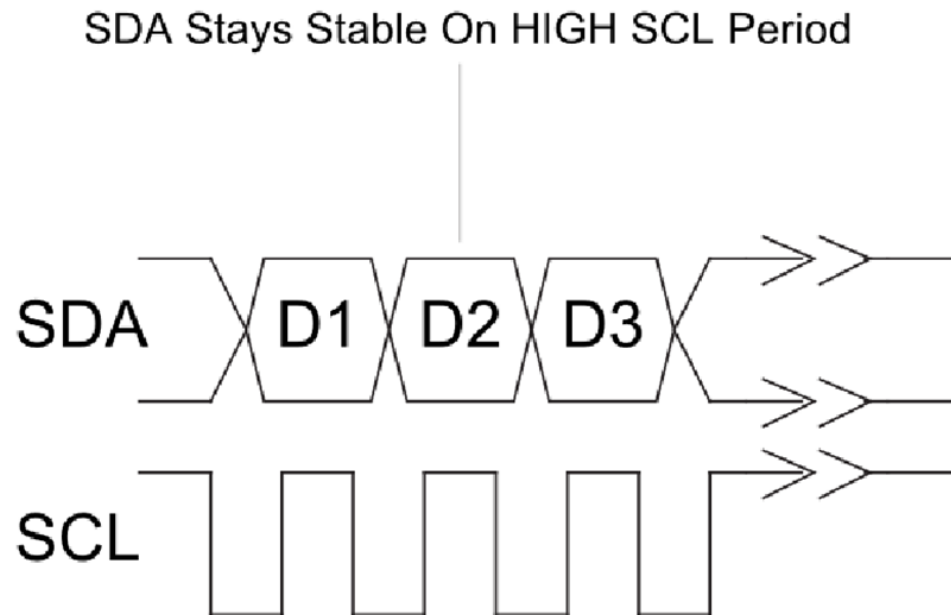
■ Data Bits

- ✓ 8-bit byte format
- ✓ Start with address byte
- ✓ each byte must follow an ACK/NACK
- ✓ the data on the SDA line **must remain stable** during a **HIGH clock** period.
- ✓ receiver reads a data bit **only** while **SCL is HIGH**
- ✓ The data line is allowed to **change** only when the **clock** signal is **LOW**.

I2C Protocol

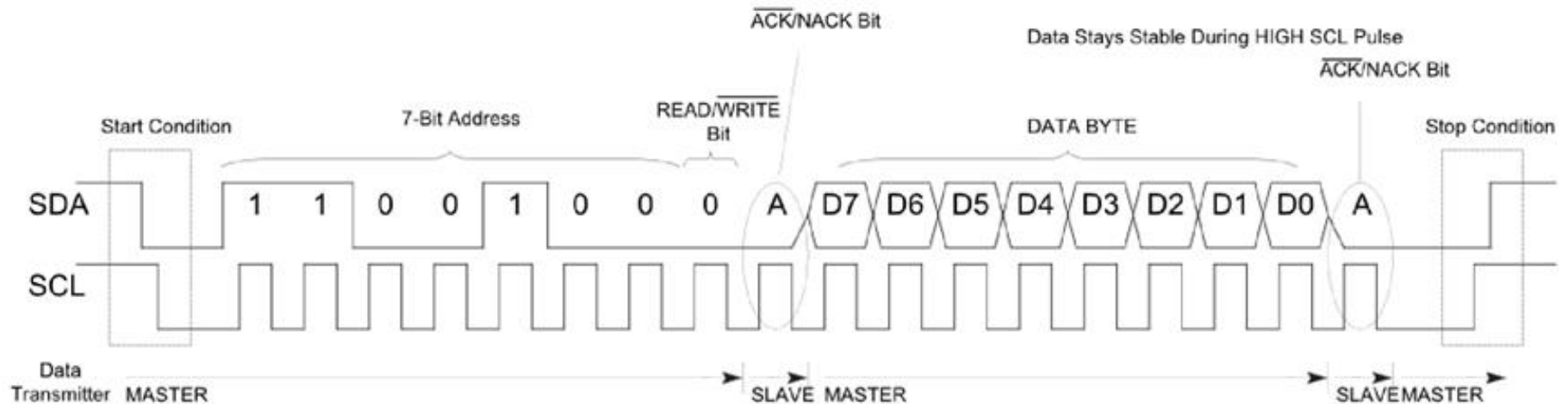
➤ How does I2C work? (cont'd)

■ Data Bits



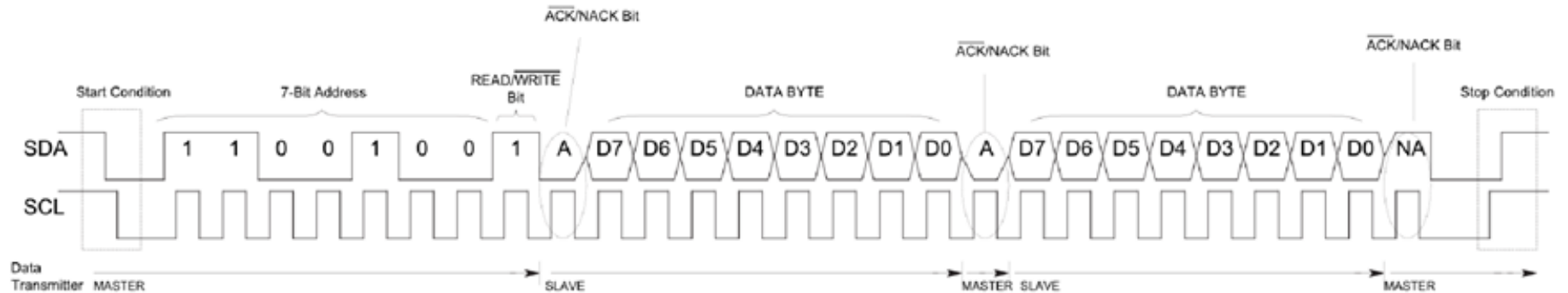
I2C Protocol

➤ Writing to a Device



I2C Protocol

➤ Reading from a Device



BUS

BUS

- The basis of BUS
- URAT Protocols
- I2C Protocols
- SPI Protocols

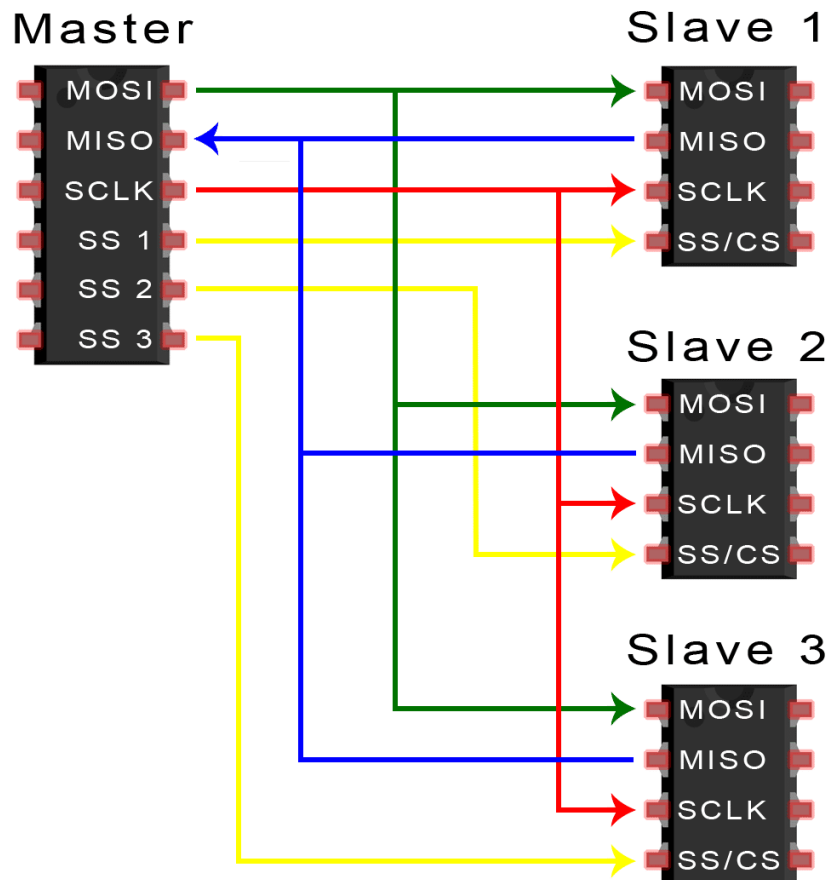
SPI Protocol

➤ What is SPI?

- Serial Peripheral Interface (SPI)
- Protocol for **Synchronous** communication in BUS system
- Follows a **master/slave** hierarchy
- One master, one/multiple slaves
- **Full-duplex**, where data can be **sent and received simultaneously**

SPI Protocol

➤ Structure of SPI system

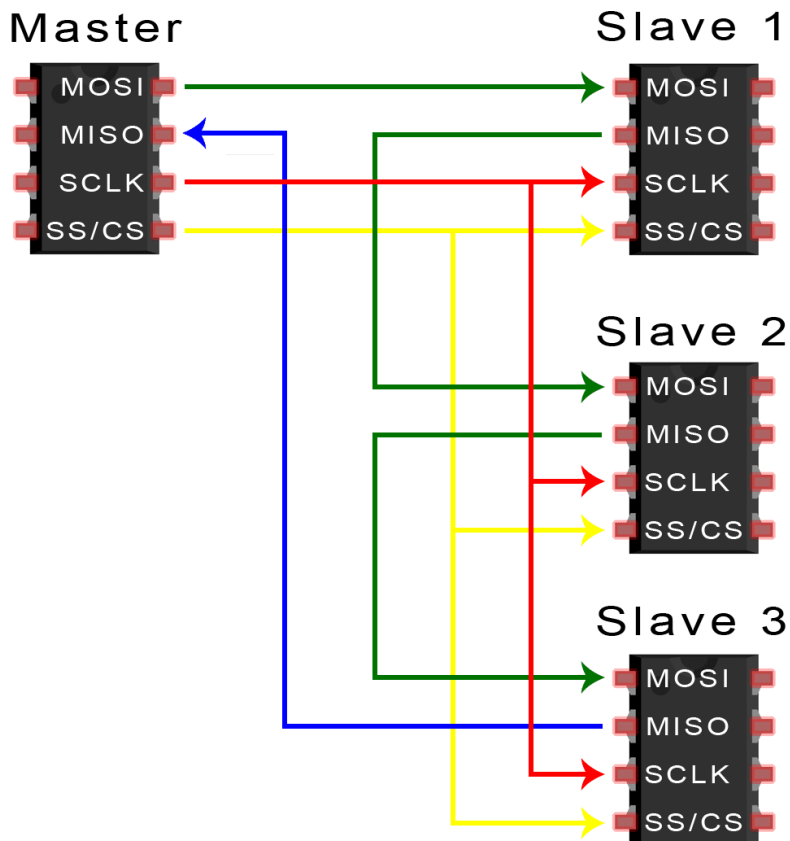


Four logical signals

- ✓ SCLK: Serial Clock, set by Master
- ✓ MOSI: master output, slave input
- ✓ MISO: master input, slave output
- ✓ SS (or CS): slave select or chip select (slaves are wired **in parallel**)

SPI Protocol

➤ Structure of SPI system (cont'd)



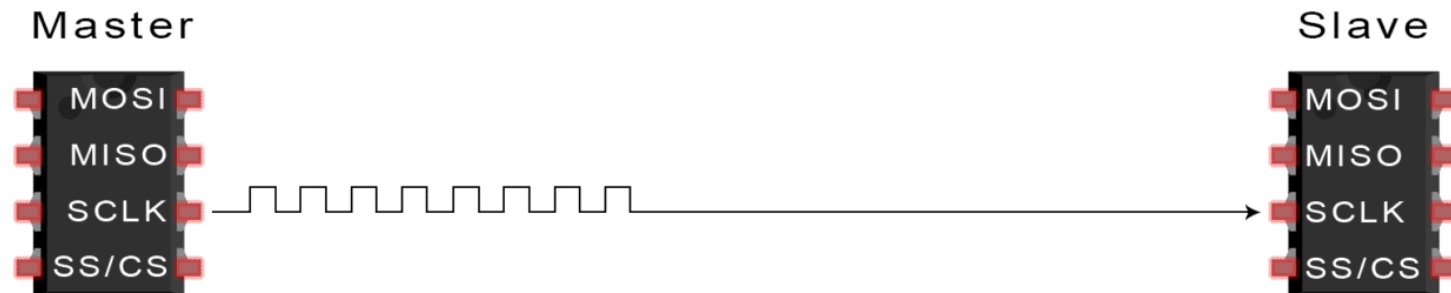
Four logical signals

- ✓ SCLK: Serial Clock, set by Master
- ✓ MOSI: master output, slave input
- ✓ MISO: master input, slave output
- ✓ SS (or CS): slave select or chip select (slaves are **daisy-chained**)

SPI Protocol

➤ How does SPI work?

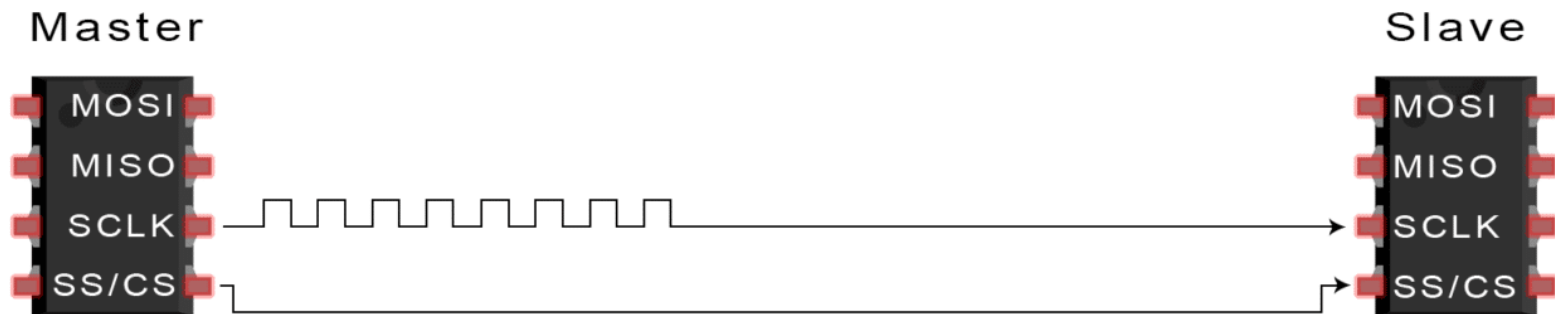
Step1. The master outputs the **clock** signal:



SPI Protocol

➤ How does SPI work(cont'd)?

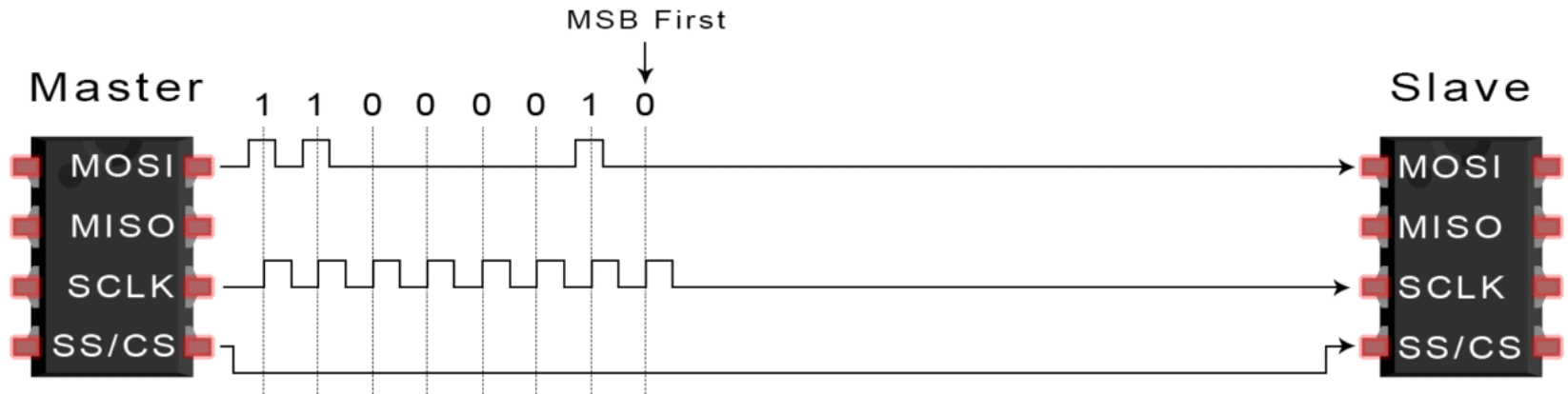
Step2: The master switches the **SS/CS** pin to a low voltage state, which activates the slave:



SPI Protocol

➤ How does SPI work(cont'd)?

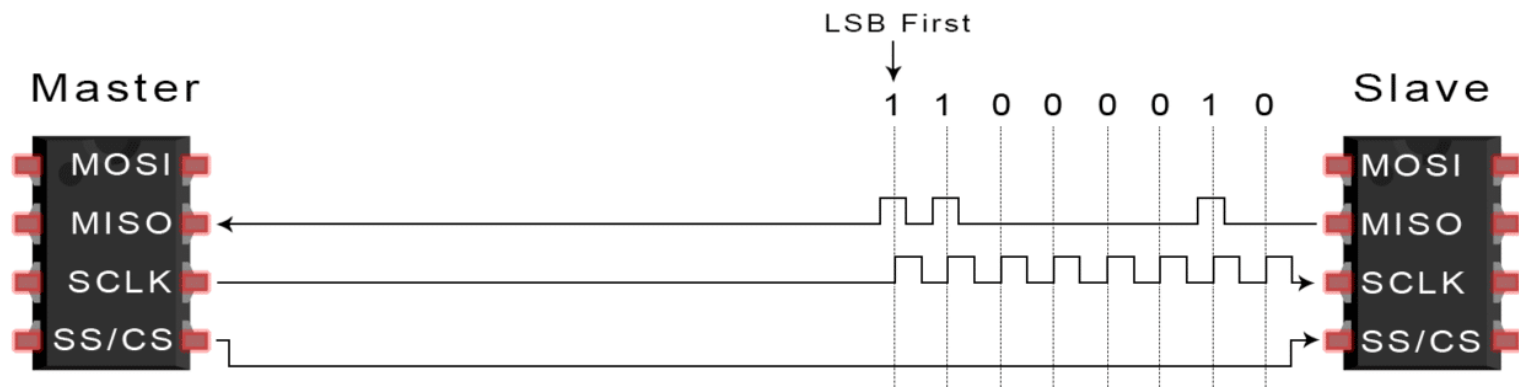
Step3: The master **sends** the data one bit at a time to the slave along the **MOSI** line. The slave reads the bits as they are received:



SPI Protocol

➤ How does SPI work(cont'd)?



Step 4: If a response is needed, the **slave returns data** one bit at a time to the master along the **MISO line**. The master reads the bits as they are received:



SPI Protocol

➤ How does SPI work(cont'd)?

- two clock parameters in SPI
 - ✓ clock polarity (**CPOL**) - sets the polarity of the clock signal during the idle state

CPOL		
0	Clock active high (off =0)	SCLK 
1	Clock active low (off=1)	SCLK 



SPI Protocol

➤ How does SPI work(cont'd)?

- two clock parameters in SPI

✓ CPOL

✓ clock phase (CPHA) - selects the clock phase

CPHA		
0	Clock out of phase with data (CPOL = 0)	 <p>SCLK</p> <p>MOSI / MISO</p>
1	Clock in phase with data (CPOL = 0)	 <p>SCLK</p> <p>MOSI / MISO</p>

SPI Protocol

➤ How does SPI work(cont'd)?

CPOL=0, CPHA=0

CPOL=0

SCLK

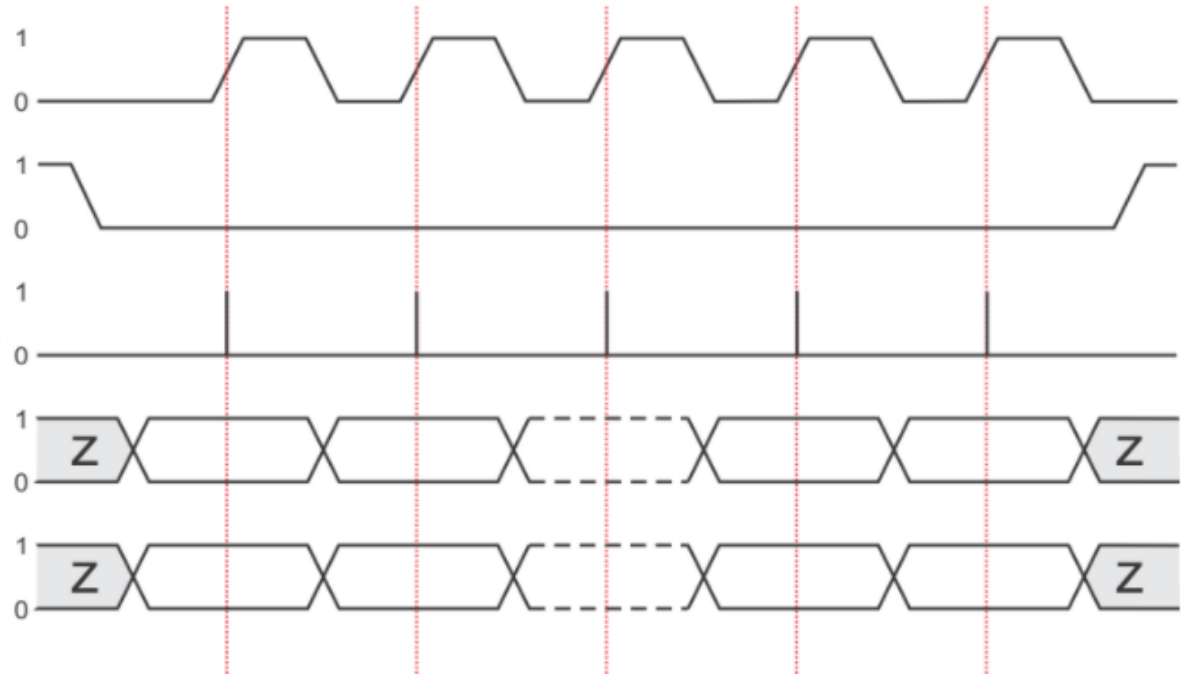
\overline{SS}

Sample

MISO

CPHA=0

MOSI

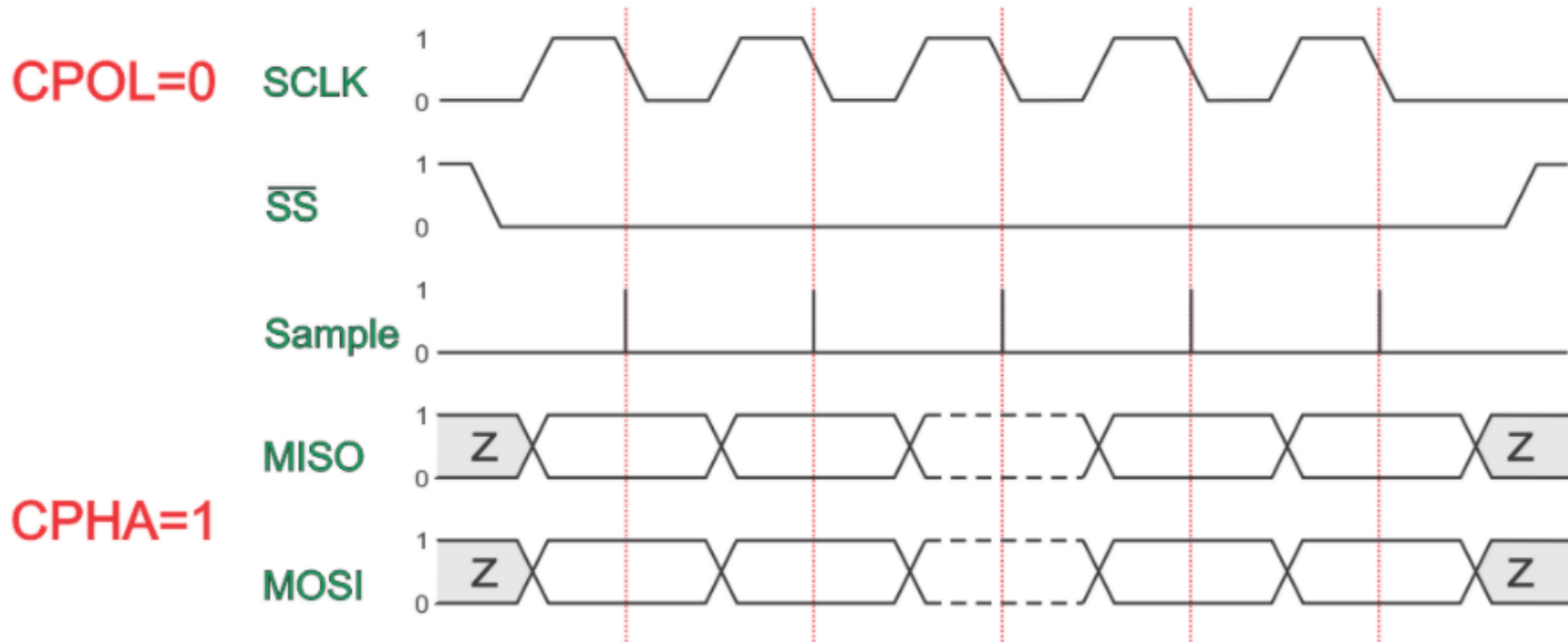


- The clock idle state is **zero**
- The data is sampled on the clock's **low-to-high (first)** transition

SPI Protocol

➤ How does SPI work(cont'd)?

CPOL=0, CPHA=1

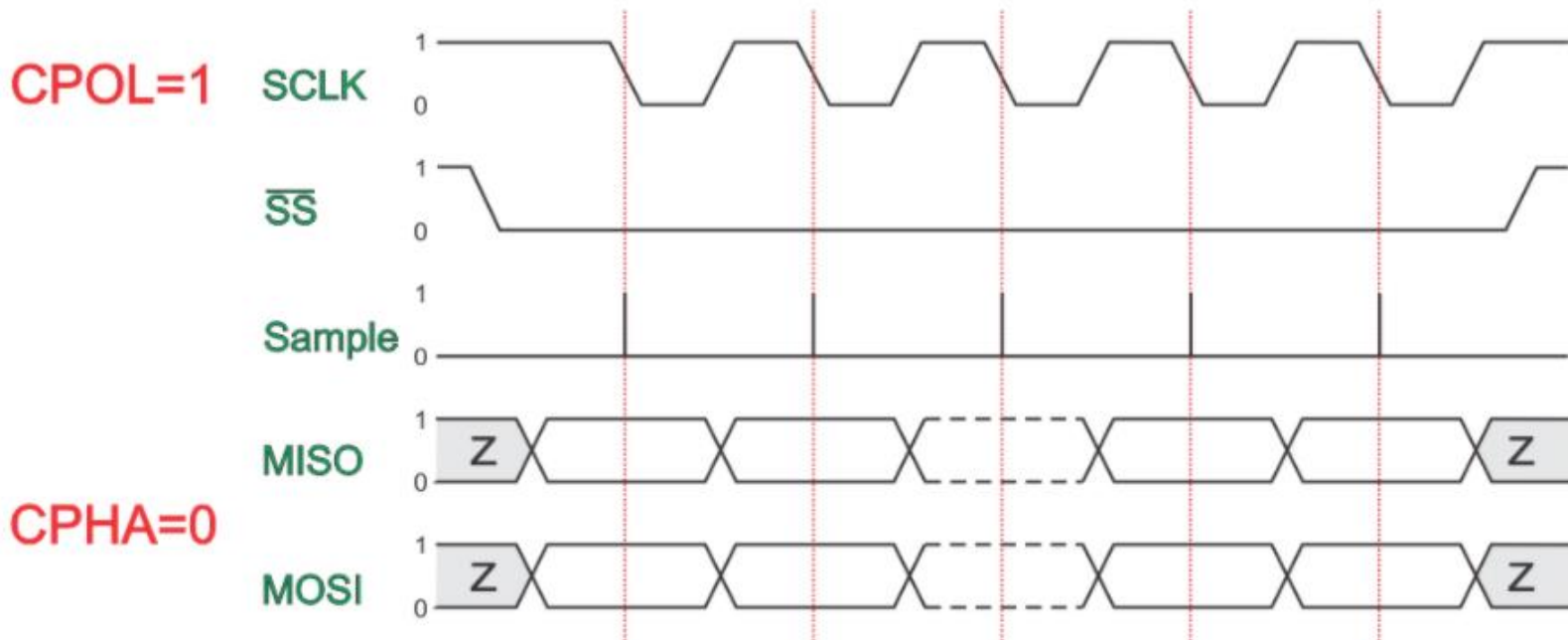


- The clock idle state is **zero**
- The data is sampled on the clock's **high-to-low (second)** transition

SPI Protocol

➤ How does SPI work(cont'd)?

CPOL=1, CPHA=0

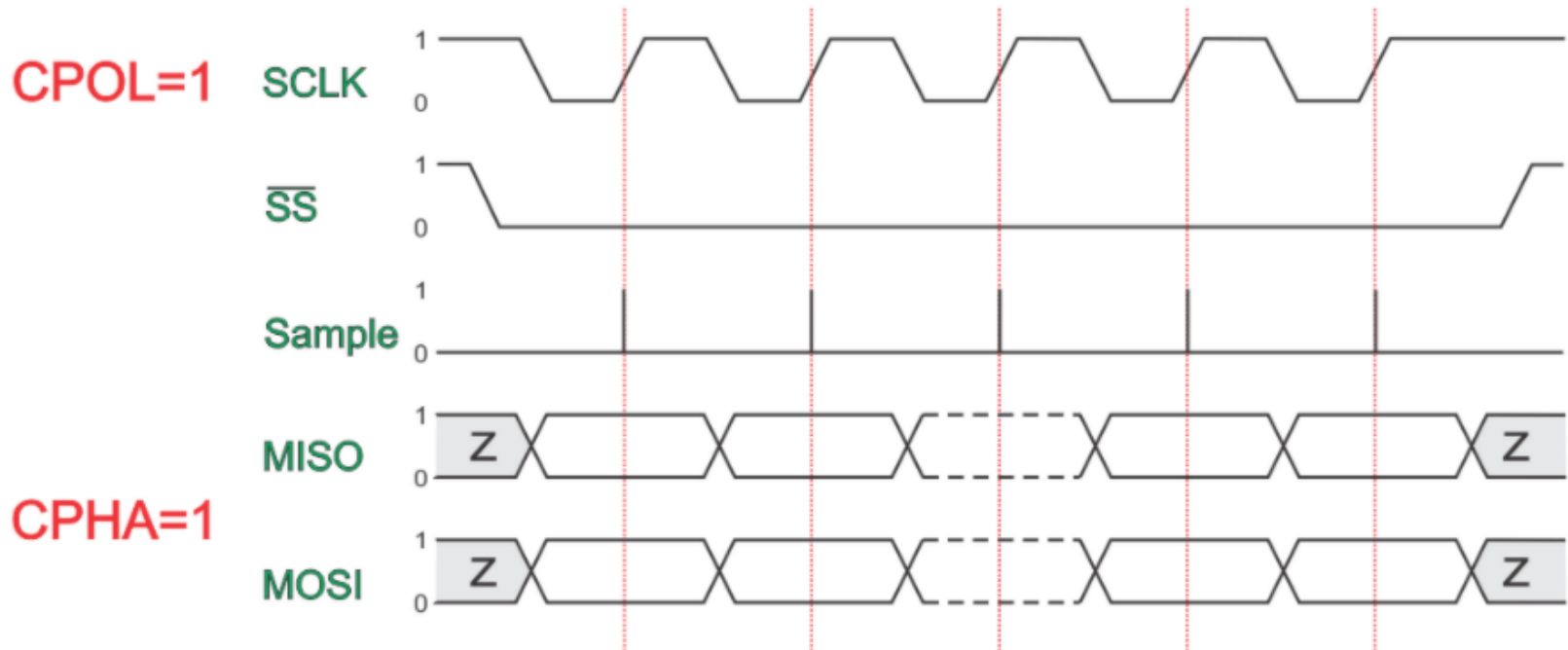


- The clock idle state is **one**
- The data is sampled on the clock's **high-to-low (first)** transition

SPI Protocol

➤ How does SPI work(cont'd)?

CPOL=1, CPHA=1



- The clock idle state is **one**
- The data is sampled on the clock's **low-to-high (second)** transition

SPI Protocol

➤ Pros and Cons

■ Advantages

- ✓ No start and stop bits, so the data can be streamed continuously without interruption
- ✓ No complicated slave addressing system like I2C
- ✓ Higher data transfer rate than I2C (almost twice as fast)
- ✓ Separate MISO and MOSI lines, so data can be sent and received at the same time (full-duplex)

SPI Protocol

➤ Pros and Cons (cont'd)

■ Disadvantages

- ✓ Uses **four wires** (I2C and UARTs use two)
- ✓ **No acknowledgement** that the data has been successfully received (I2C has this)
- ✓ **No form of error checking** like the parity bit in UART
- ✓ Only allows for a **single** master



University
of Glasgow

THANKS !

**WORLD
CHANGING
GLASGOW**

