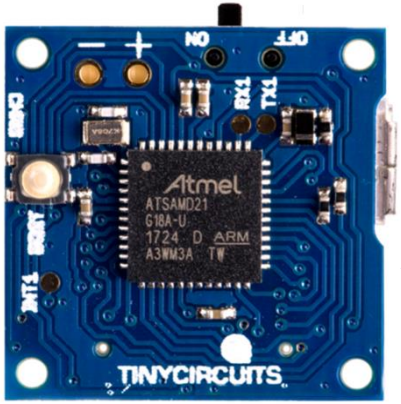
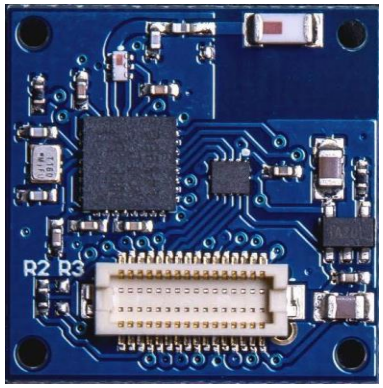
CSE190 Winter 2025



Lecture 11

Serial Busses (cont)



Wireless Embedded Systems

Aaron Schulman

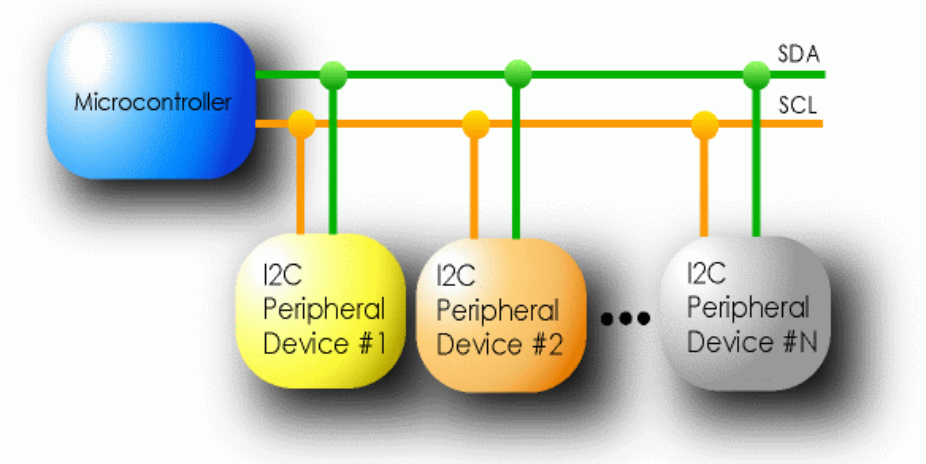
I2C bus (in our project)

- Communication with the accelerometer
  - Read acceleration values and configure interrupts
- **Pros**
  - Two wires bus that can connect multiple peripherals with the MCU
- **Cons**
  - Overhead is significantly higher, and bus is slower

I2C Details

- Two lines
  - Serial data line (SDA)
  - Serial clock line (SCL)

- Only two wires for connecting multiple devices

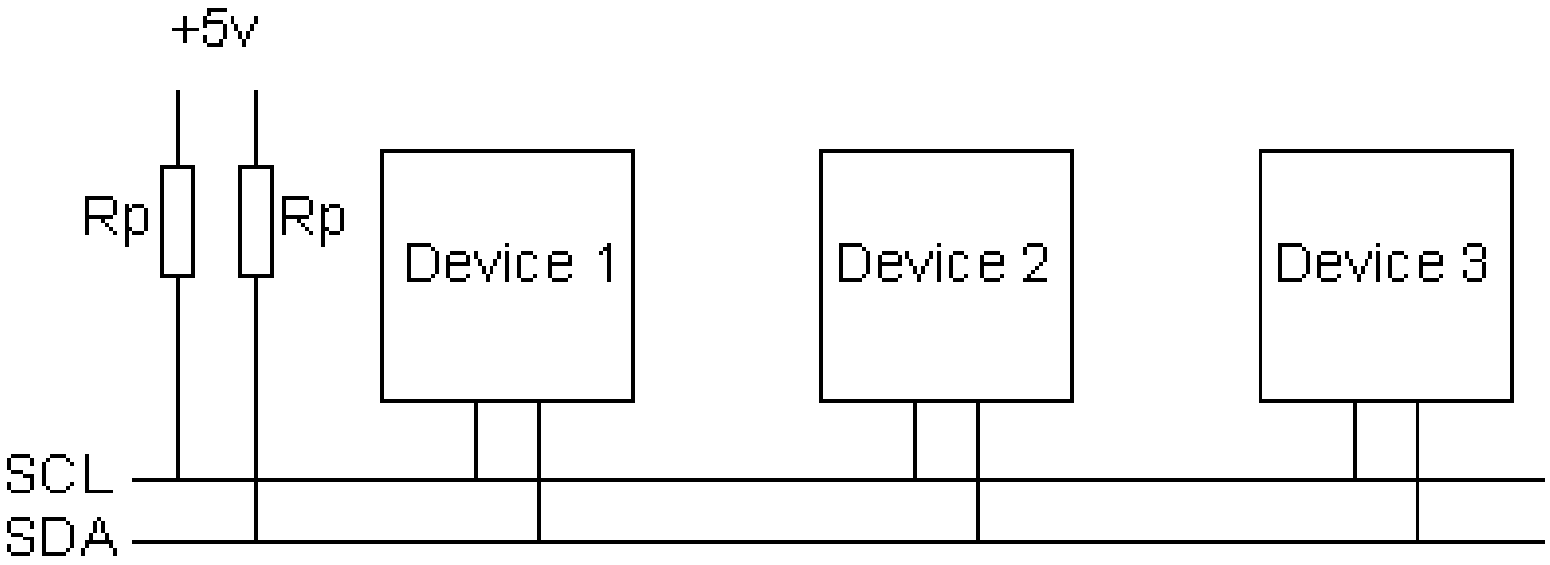


I2C Details

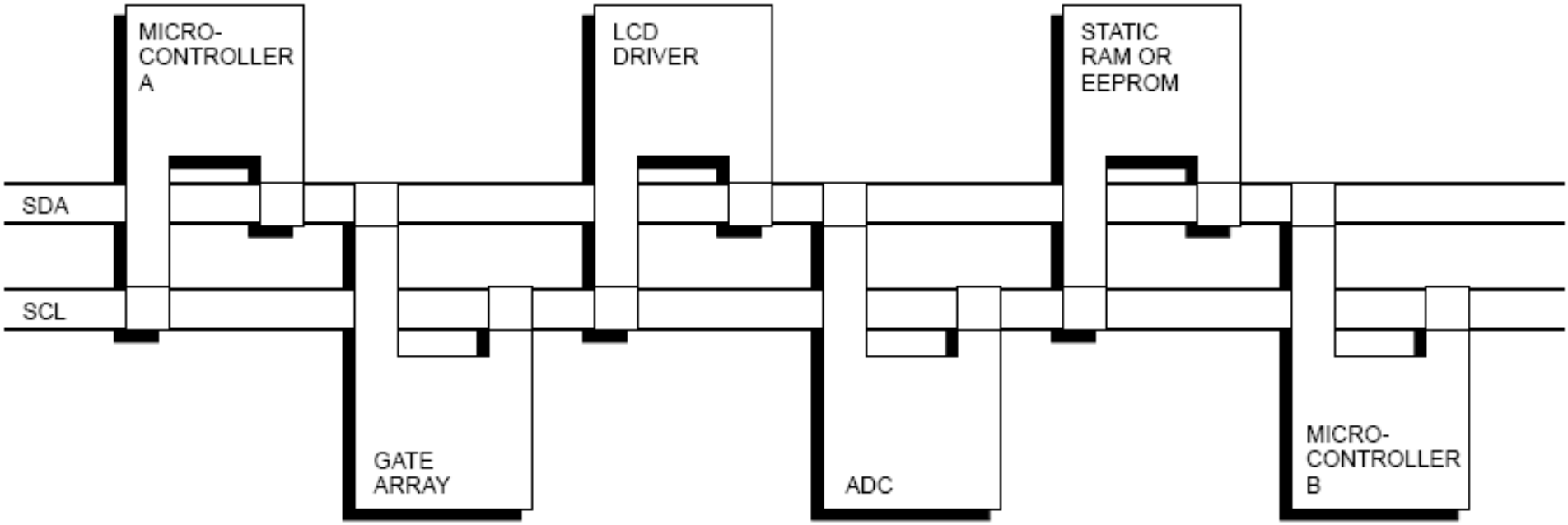
- Each I2C device recognized by a unique address
- Each I2C device can be either a transmitter or receiver
- I2C devices can be primarys or secondarys for a data transfer
  - primary (usually a microcontroller): Initiates a data transfer on the bus, generates the clock signals to permit that transfer, and terminates the transfer
  - secondary: Any device addressed by the primary at that time

How can any device transfer or receive on the same two wires?

- Pull ups and high-impedance mode pins
  - Wires default to being “high”, any device can make a wire go “low”.
  - This is super clever. UART can’t do this, why?



I2C-Connected System



Example I2C-connected system with two microcontrollers

(Source: I2C Specification, Philips)

Primary/Secondary Relationships

- Who is the primary?
  - primary-transmitters
  - primary-receivers

- Suppose microcontroller A wants to send information to microcontroller B
  - A (primary) addresses B (secondary)
  - A (primary-transmitter), sends data to B (secondary-receiver)
  - A terminates the transfer.

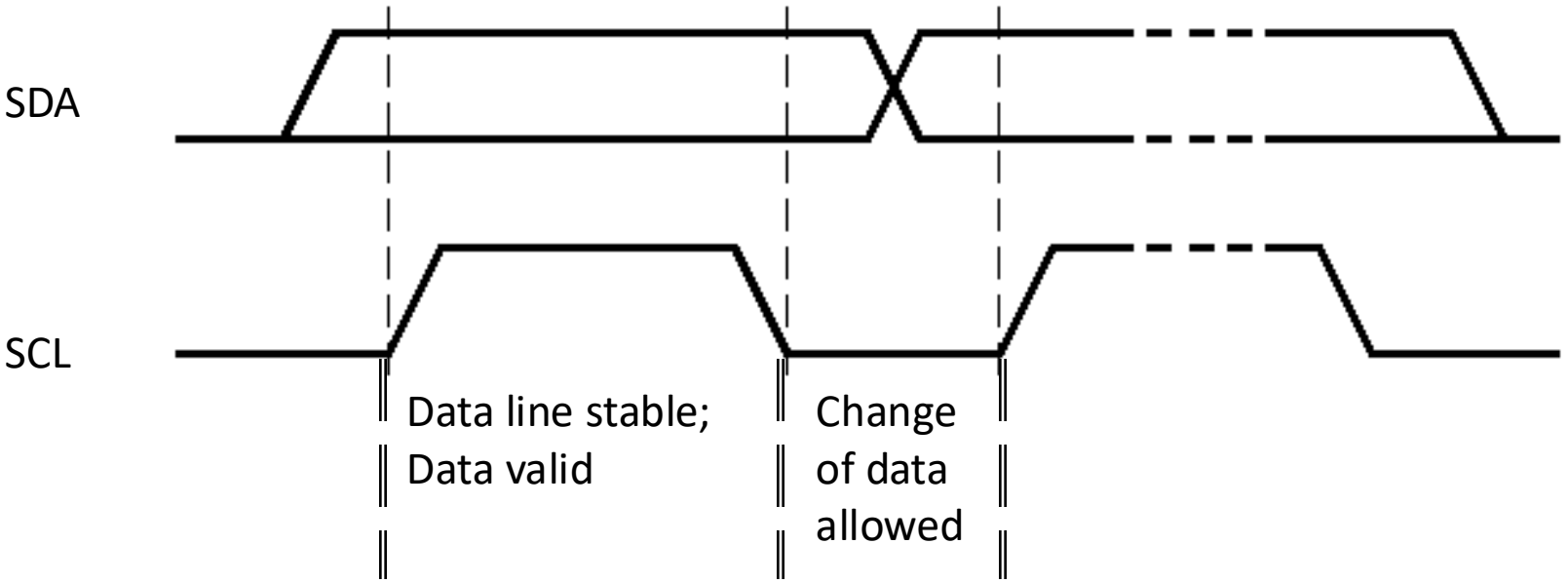
- If microcontroller A wants to receive information from microcontroller B
  - A (primary) addresses microcontroller B (secondary)
  - A (primary-receiver) receives data from B (secondary-transmitter)
  - A terminates the transfer

- In both cases, the primary (microcontroller A) generates the timing and terminates the transfer

- Terminology varies: Primary/Secondary or Master/Slave etc. (Depending on manual)

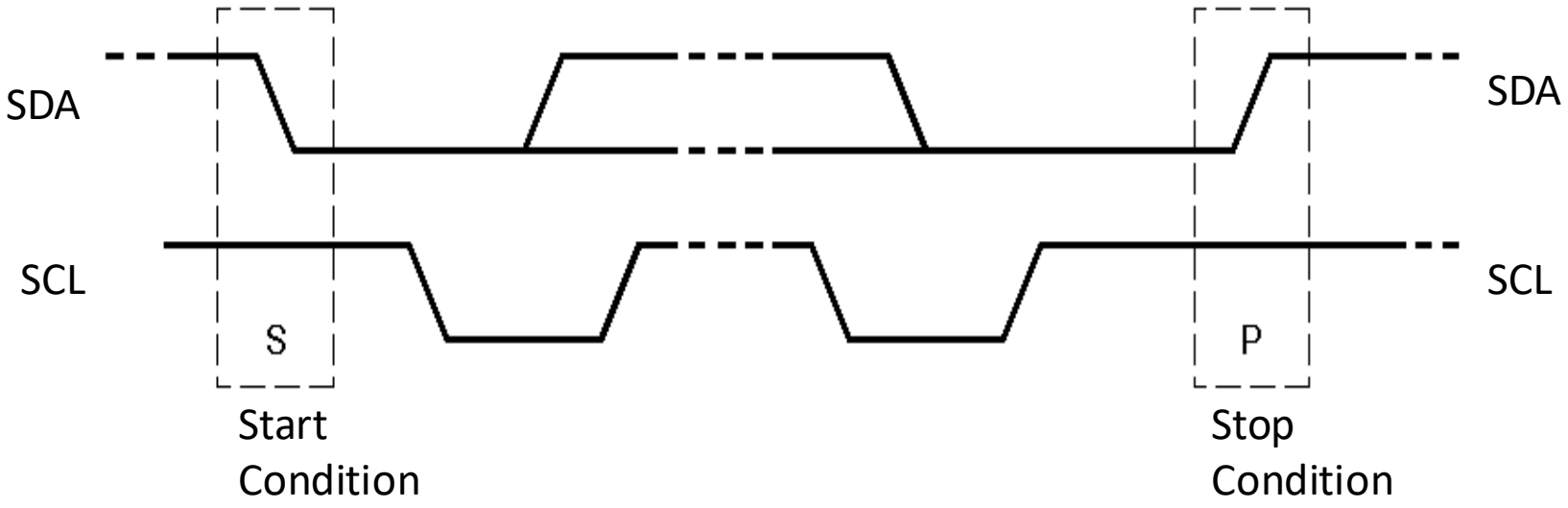
Bit Transfer on the I^2C Bus

- In normal data transfer, the data line only changes state when the clock is low



Start and Stop Conditions

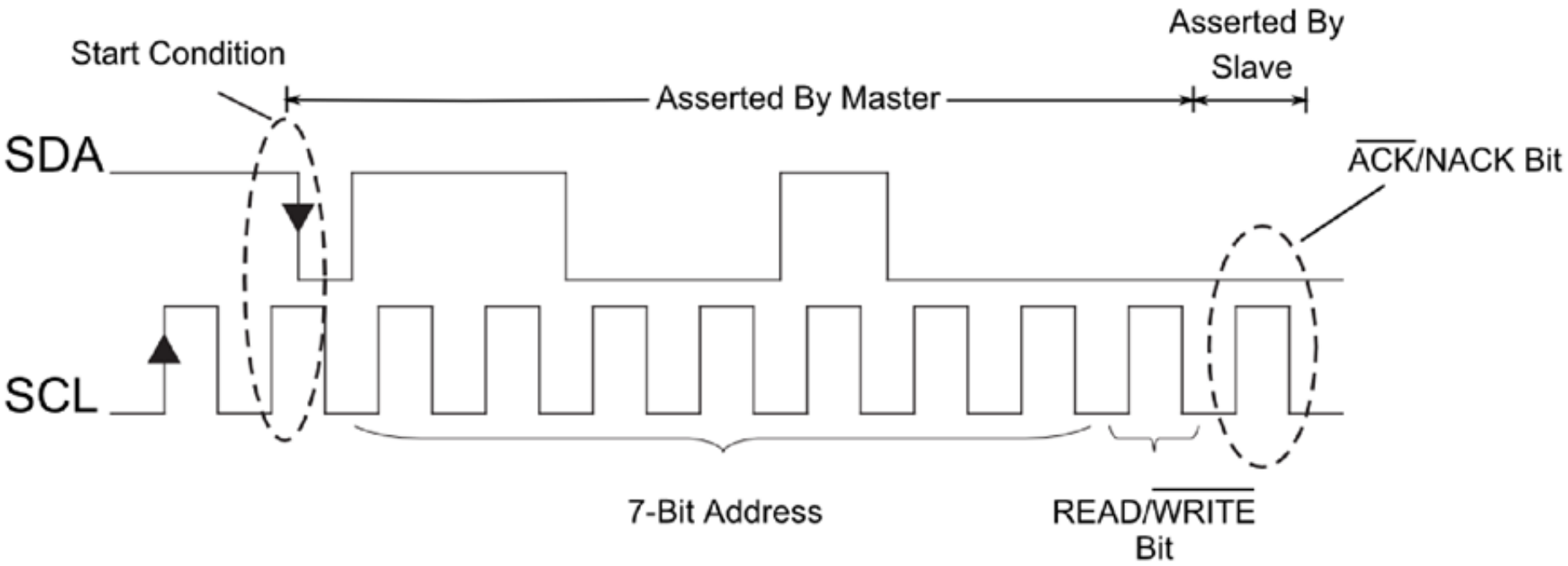
- A transition of the data line while the clock line is high is defined as either a start or a stop condition.
- Both start and stop conditions are generated by the bus primary
- The bus is considered busy after a start condition, until a stop condition occurs



I^2C Addressing (Multiple devices)

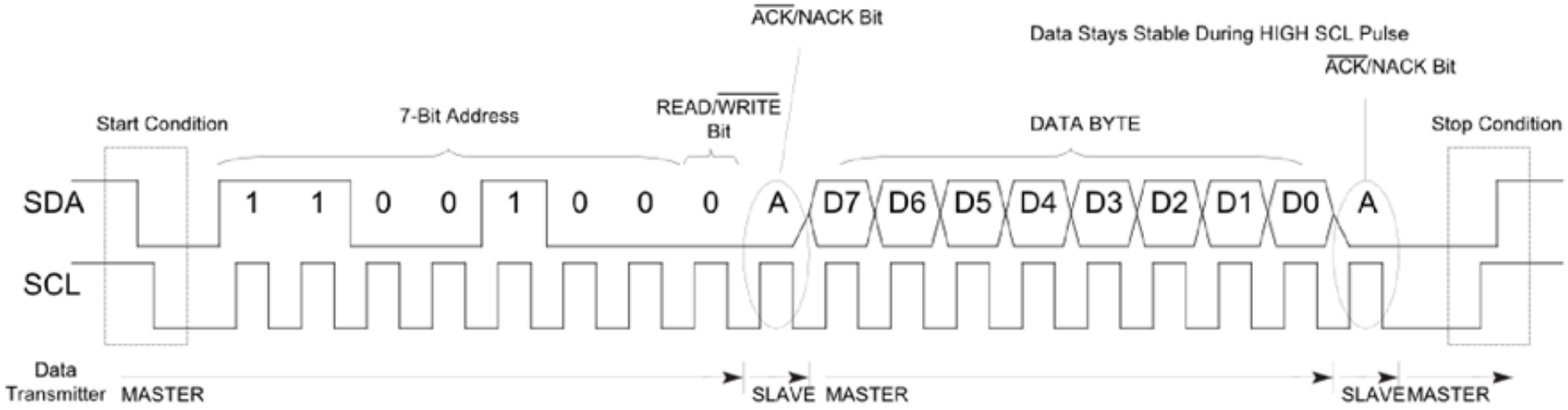
- Each node has a unique 7 (or 10) bit address
- Peripherals often have fixed and programmable address portions
- Addresses starting with 0000 or 1111 have special functions:-
  - 0000000 Is a General Call Address
  - 0000001 Is a Null (CBUS) Address
  - 1111XXX Address Extension
  - 1111111 Address Extension – Next Bytes are the Actual Address

Beginning of I2C Transaction



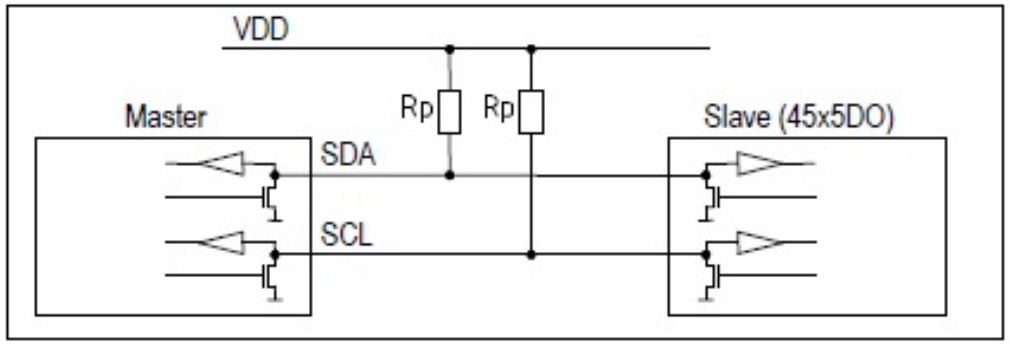
Source: Analog

Full I2C Transaction

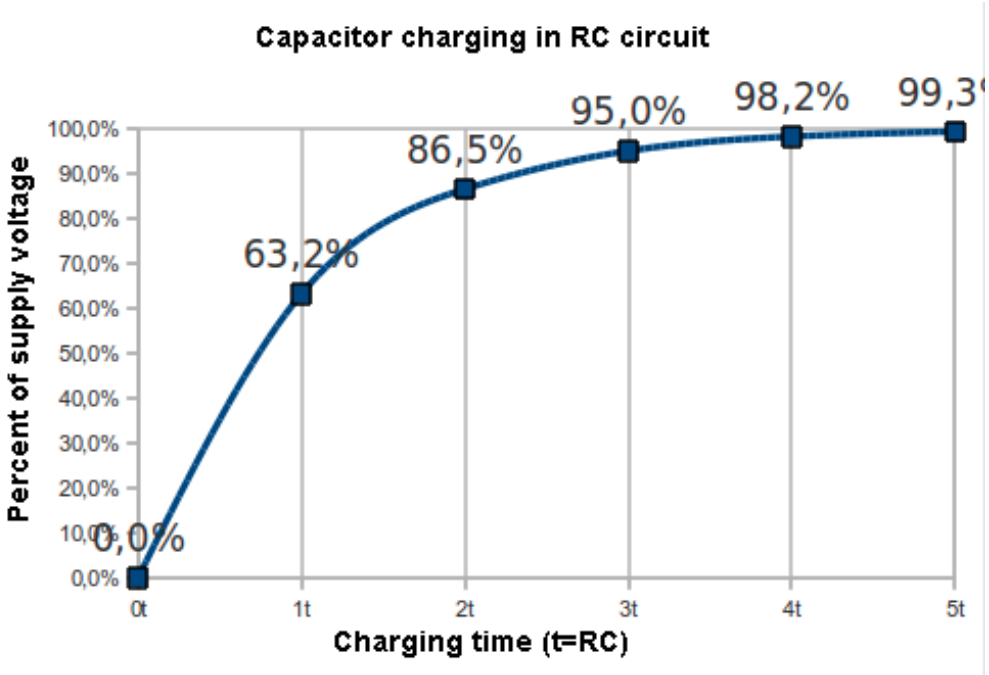


Source: Analog

How fast can I2C run?



- How fast can you run it?
- Assumptions
  - 0's are driven
  - 1's are "pulled up"
- Some working figures
  - R_p = 10 kΩ
  - C_{cap} = 100 pF
  - V_{DD} = 5 V
  - V_{in_high} = 3.5 V
- Recall for RC circuit
  - $V_{cap}(t) = V_{DD}(1-e^{-t/\tau})$
  - Where $\tau = RC$

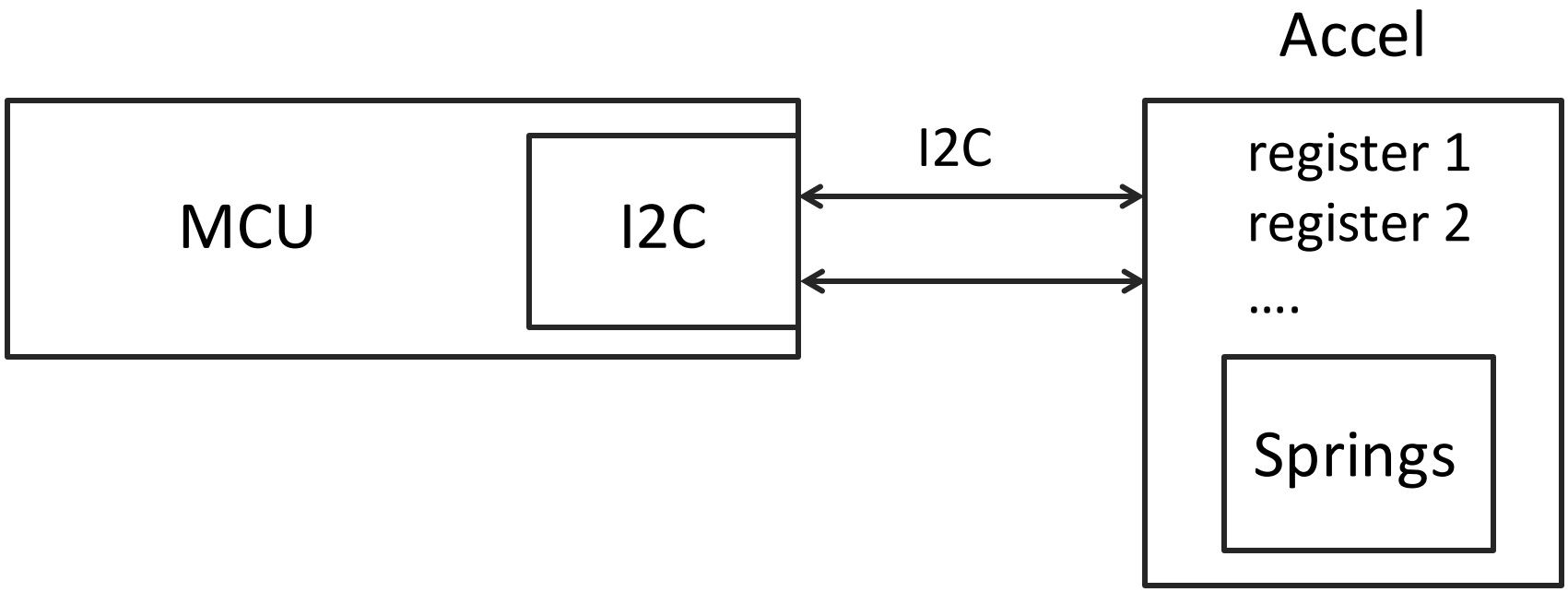


Practically I2C can do at most 400kbps

Exercise: Bus bit rate vs Useful data rate

- An I2C “transactions” involves the following bits
  - <S><A6:A0><R/W><A><D7:D0><A><F>
- Which of these actually carries useful data?
  - <S><A6:A0><R/W><A><D7:D0><A><F>
- So, if a bus runs at 400 kHz
  - What is the clock period?
  - What is the data throughput (i.e. data-bits/second)?
  - What is the bus “efficiency”?

How to operate the accelerometer?



https://www.youtube.com/watch?v=eqZgxR6eRjo