COPYRIGHT

DISCLAIMER

This content is provided without warranty or representation of any kind. The use of the content is entirely at your own risk and Stellenbosch University (SU) will have no liability directly or indirectly as a result of this content.

The content must not be assumed to provide complete coverage of the particular study material. Content may be removed or changed without notice.

The video is of a recording with very limited post-recording editing. The video is intended for use only by SU students enrolled in the particular module.

UNIVERSITEIT
iYUNIVESITHI
STELLENBOSCH
UNIVERSITY

100
1918 · 2018

forward together · saam vorentoe · masiye phambili

Computer Systems / Rekenaarstelsels 245 - 2020

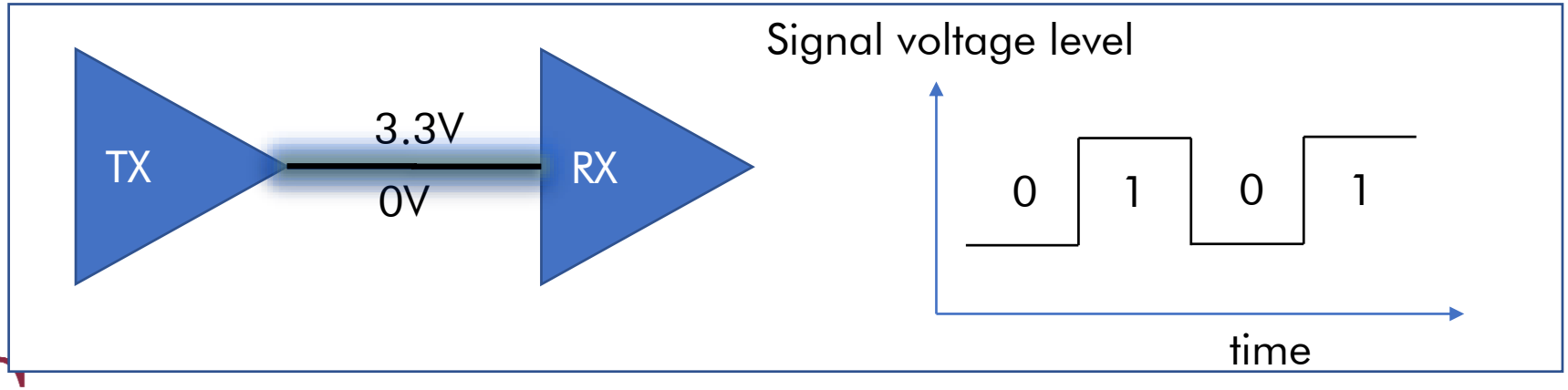
Lecture 15

Serial Communication Introduction/ Seriële Kommunikasie Inleiding

Dr Rensu Theart & Dr Lourens Visagie

What is serial communication?
Wat is seriële kommunikasie?

- **Serial communication**: Transmitting data one bit at a time (sequentially) over a communications channel.
- Voltage levels on a shared connection are varied over time to indicate binary information
- The connection is usually implemented as a physical wire, or a PCB track
- At the most basic:
  - Transmitter or sender (TX) sends the data
  - Receiver (RX) receives the data
  - Signaling all happens on a single connection



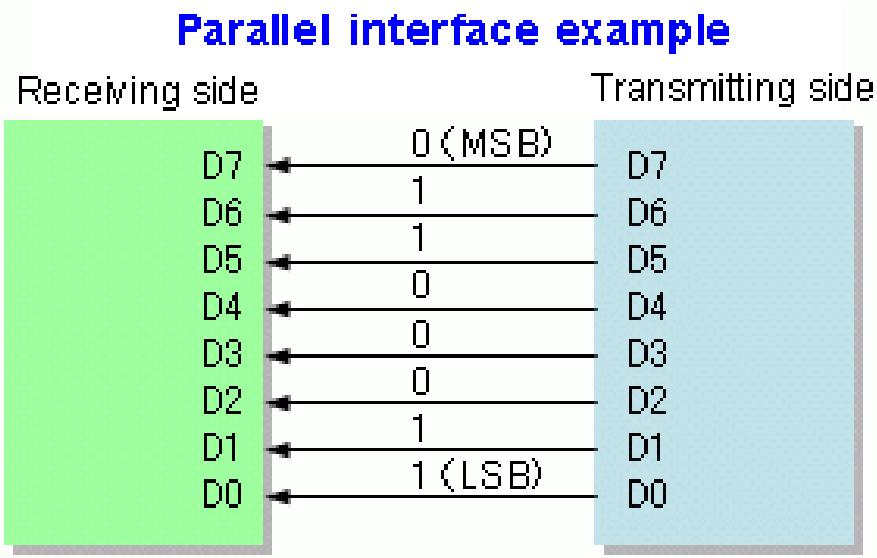
What is serial communication?
Wat is seriële kommunikasie?

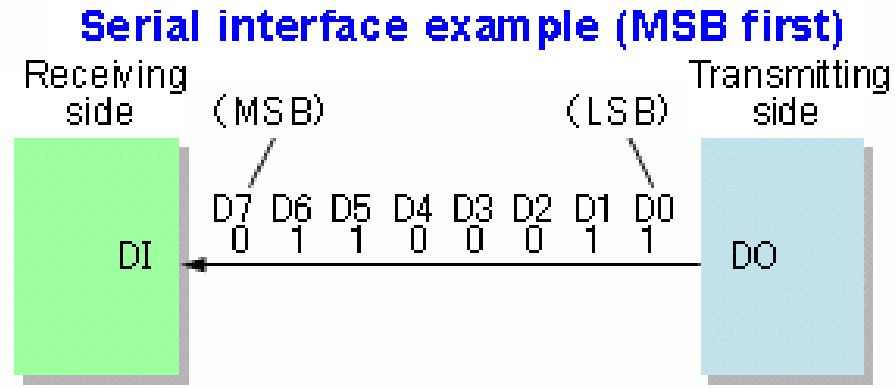
- **Serial communication**: Transmitting data one bit at a time (sequentially) over a communications channel.
- Voltage levels on a shared connection are varied over time to indicate binary information
- The connection is usually implemented as a physical wire, or a PCB track
- At the most basic:
  - Transmitter or sender (TX) sends the data
  - Receiver (RX) receives the data
  - Signaling all happens on a single connection
- But it is usually a bit more complicated
  - Transmitter and receiver can swop roles
  - Some communication schemes allow multiple devices to use the same connection – then it becomes a communications bus
  - Some communication schemes have additional signals – improve speed and reliability

What is serial communication?
Wat is seriële kommunikasie?

Serial vs. Parallel communication

- **Serial** communications: Single line/signal/wire/channel that is varied over time to transfer multiple bits of information
- **Parallel** communications: Multiple lines/signals/wires/channels (also varying over time) that can transfer multiple bits of information at a single time instant
- Serial communication takes longer to send the same data, but parallel communications have more signals/cables

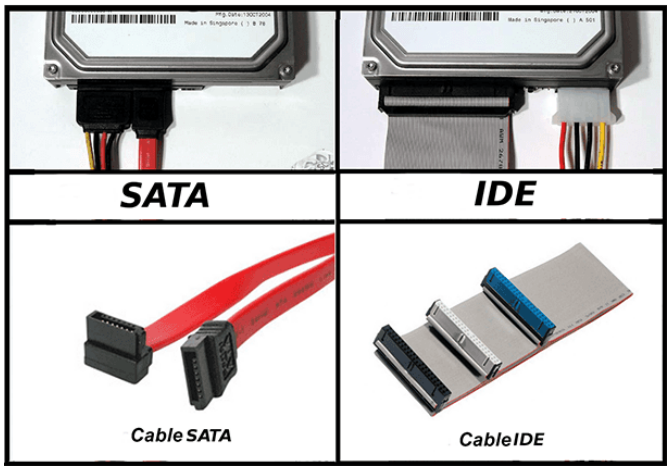




What is serial communication?
Wat is seriële kommunikasie?

Serial vs. Parallel communication

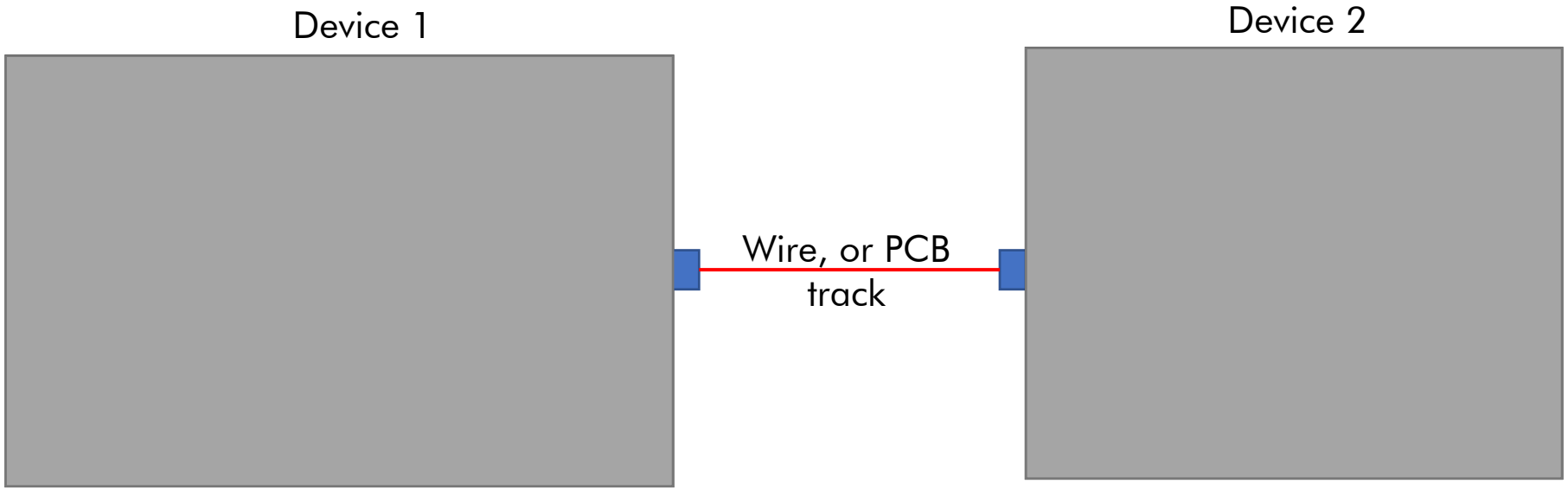
- Example: Hard drives – IDE (parallel) vs. SATA (serial)



- Nowadays, the tendency is to use serial communication across physical cables (USB, HDMI)
- Parallel communication is found inside the microcontroller (system bus), or as copper tracks on a PC mainboard or other PCBs – typical for external memory interfaces, or camera interfaces

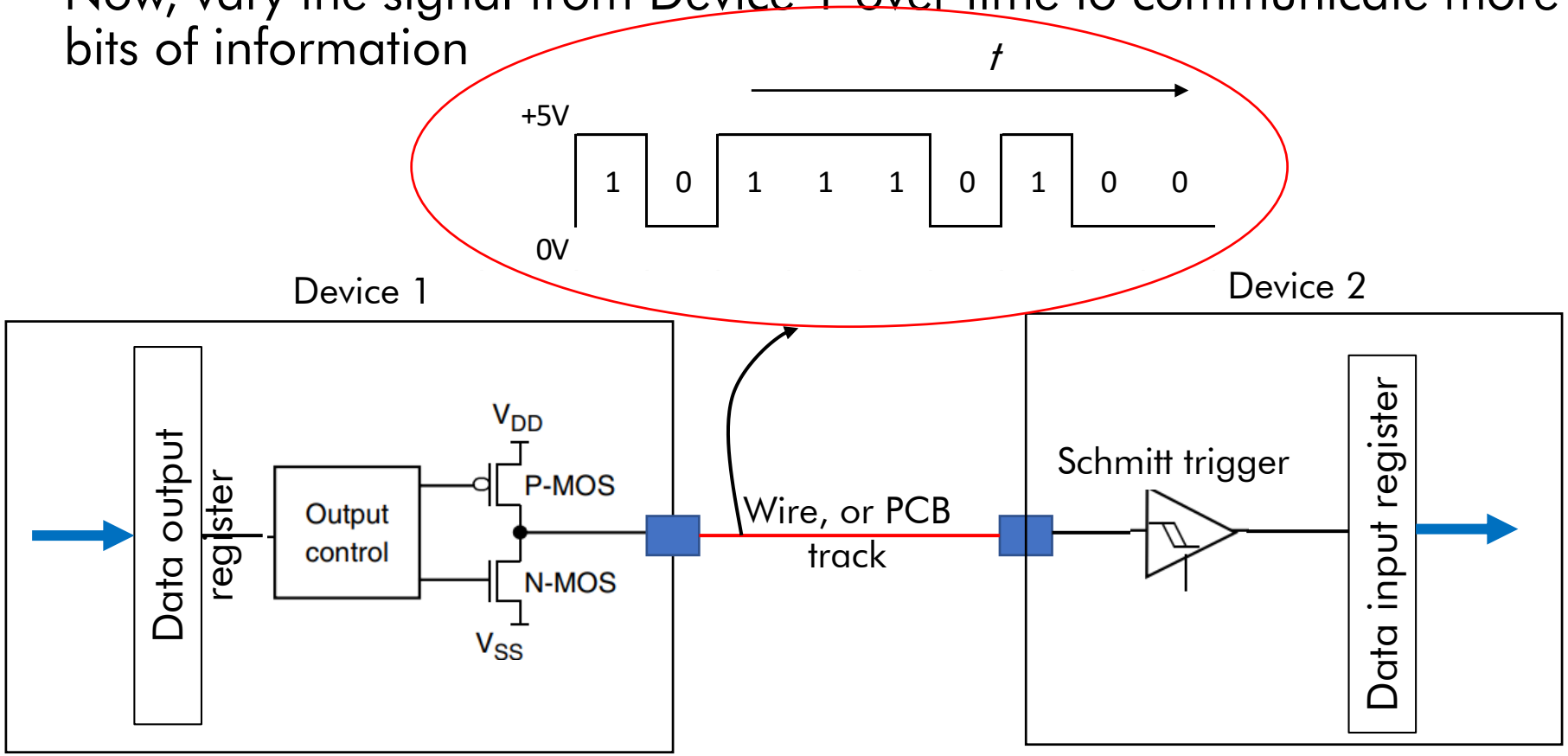
From GPIO to inter-device communication
Vanaf basiese intree/uittree na kommunikasie

- On device 1, use a pin configured as an output – this device (microcontroller) will drive the signal level on the output to a zero or one, depending on what the program for that microcontroller does
- On device 2, use a pin configured as input – this microcontroller will not try to control the signal level, but sense the signal level
- Connect the two pins together. Now device 2 can sense the signal that device 1 is outputting. (Device 1 is transferring information to device 2)



From GPIO to inter-device communication
Vanaf basiese intree/uittree na kommunikasie

- Using this strategy, we can convey 1-bit of information at one instant in time
- Now, vary the signal from Device 1 over time to communicate more bits of information



From GPIO to inter-device communication
Vanaf basiese intree/uittree na kommunikasie

Questions…

1.

How does device 2 know when the next bit of information is going to be valid? (this is called **synchronization**)

2.

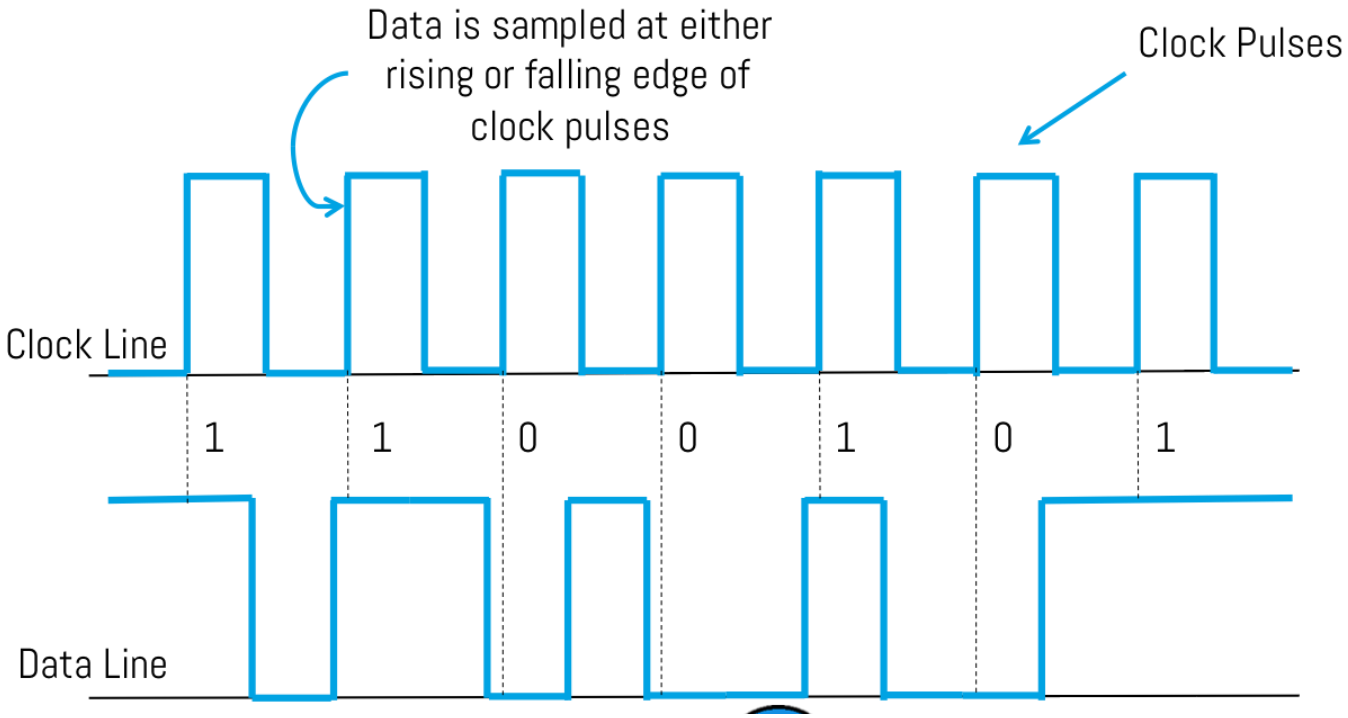
So far this is only one-way communication. How does device 2 reply information to device 1 (**unidirectional** vs. **bi-directional** communication)?

Synchronous vs. Asynchronous communication
Sinkrone en asinkrone kommunikasie

- **Communication synchronisation** – means with which the receiving device can determine when, in the time-varying data stream, one bit of information begins and the other ends
- Serial communication can be classified as synchronous or asynchronous

Synchronous vs. Asynchronous communication
Sinkrone en asinkrone kommunikasie

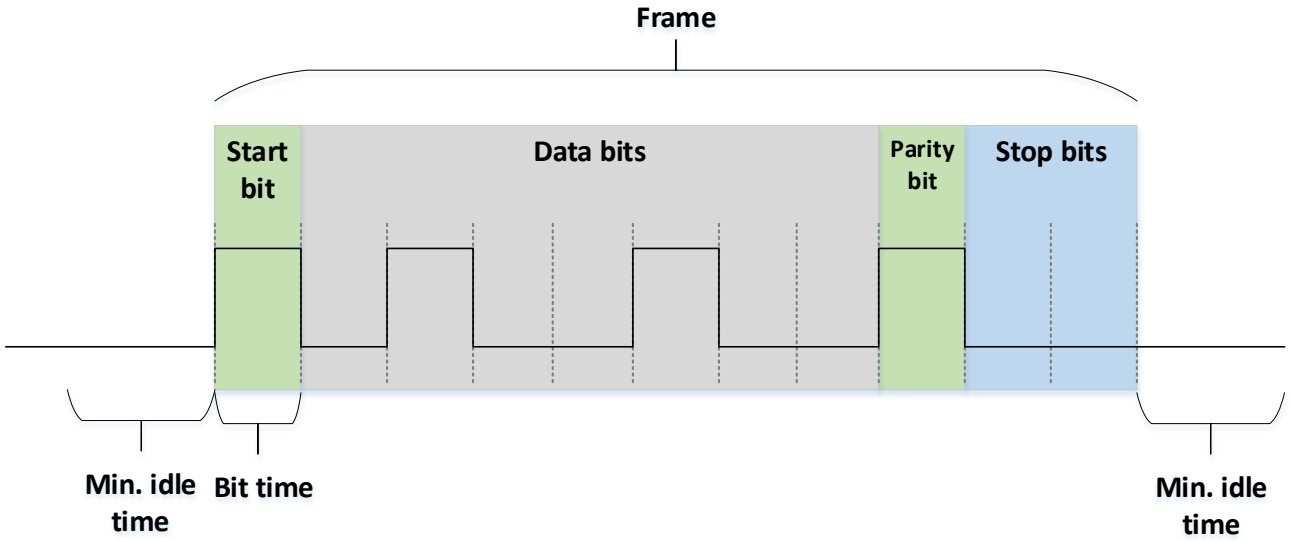
- **Synchronous** communication – separate clock signal is used to notify receiver when the next bit can be read



- **Pros**: Simple implementation. Allows for fast data transfer
- **Cons**: One more signal or wire is needed

Synchronous vs. Asynchronous communication
Sinkrone en asinkrone kommunikasie

- **Asynchronous** communication – the receiver figures it out on its own, based on
  - A known (expected signal rate)
  - The transitions in the data stream (when 0 changes to 1, or 1 to 0)
  - Special markers in the signal (Start bit, stop bit)



- **Pros**: Still only a single signal
- **Cons**: Implementation is more complex, slower data speeds (compared to synchronous)

Bi-directional communication
Bi-direksionele kommunikasie

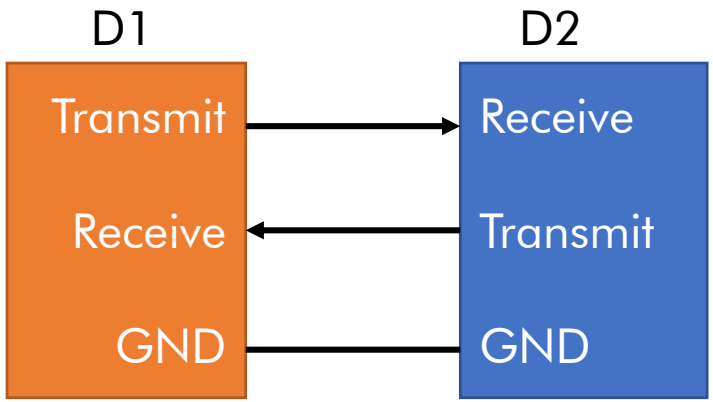
- Communication in one direction only has limited use
- **Bi-directional communication**: information can flow in both directions
- Some definitions:

Simplex	Half-duplex	Full-duplex
Communication in one direction only	Bi-directional communication, but only in one direction at a time	Simultaneous bi-directional communication

Point-to-point	Bus
Only two devices with dedicated communications channel between them	The same communication channel is shared by multiple devices

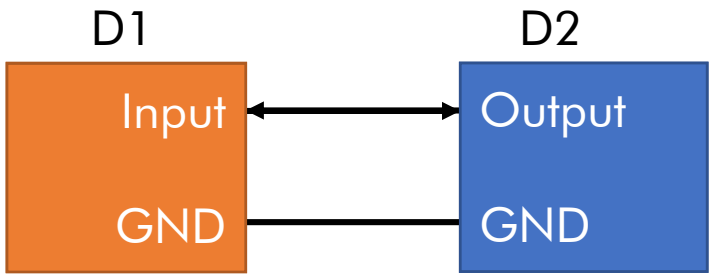
Bi-directional communication
Bi-direksionele kommunikasie

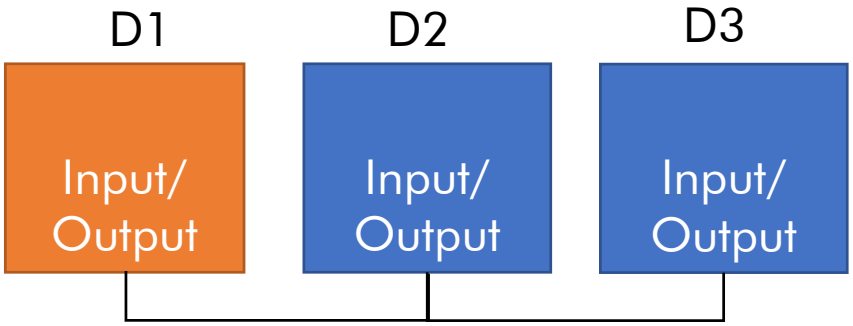
- How then do we achieve bi-directional communication?
- Option 1: Use 2 signals – one for sending data from Device 1 to Device 2, and another one to send from D2 to D1
- This scheme is a full-duplex, point-to-point communications link
- Make sure about signal direction: Signals are usually labelled Transmit (TX) or Receive (RX), but relative to which device? – check the data sheet!
- (Need a common reference voltage for the signals - connect GND as well)



Bi-directional communication
Bi-direksionele kommunikasie

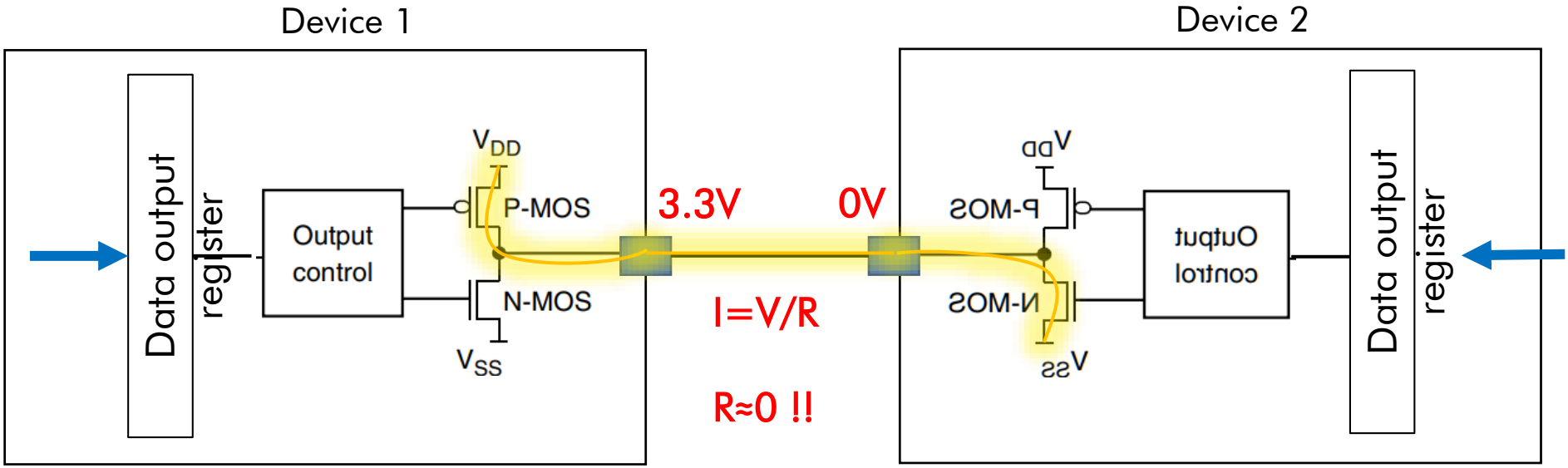
- <u>Option 2</u>: Another way – use the same physical wire/track/signal, but set device 1 as input/output, and device 2 as output/input whenever communication direction has to change
- Now we need a way to synchronise which device is allowed to control the signal
- This idea can be extended to a **communications bus** – more than two devices can use the same "line" to communicate.
- Usually in a bus communication architecture, one device will control the synchronization (the **master** device) and other devices (**slave** devices) will only communicate when allowed by the master





Bi-directional communication
Bi-direksionele kommunikasie

- Bus Contention: General term for two devices trying to use the bus at the same time
- At hardware level, this can be a problem: Let's say both devices are configured as output, device one outputs a logical 1, and device 2 outputs a logical 0



- This will cause a short-circuit! There is a limit to how much current the MOSFETs can supply and sink. The GPIO port of the microcontroller will get damaged
- The bus communication scheme has to prevent such a situation from occuring

Dedicated Serial Communication Hardware
Toegewyde seriële kommunikasie hardeware

- How would the program look that communicates using GPIO? -> LOTS of writing to memory mapped port registers. Timers. Interrupts
- At communication rate of 1 Mbps (1,000,000 bits per second), you will get 1 million interrupts per second! With a 10MHz clock and 1 machine instruction per clock cycle, you can only execute 10 machine instructions in the ISR
- The answer is to use special hardware that takes care of lower-level communications (synchronization, timing, data formatting and signal levels) – frees up the CPU so that it can perform other higher-level tasks
- Only interrupt processor for higher-level interaction, i.e. interrupt after transmission ended, or when data is arriving
- Sometimes removes the need for CPU interaction during the transfer altogether (Direct Memory Access – DMA)

- Communication will only work if both sets of specialised hardware follow the same rules for synchronisation, bi-directional transfers and signal levels. This leads to standardisation of a communications interface
- Standard serial communications interfaces: UART, I^2C, SPI, CAN, Ethernet, Serial ATA, USB, etc.
- Some of these are so common, most microcontrollers have such hardware embedded (UART, I^2C, SPI)

Standard Serial Communication
Standaard Seriële Kommunikasie

	Half/Full-duplex	Bus/point-to-point	Synchronous/ Asynchronous
UART (Universal Asynchronous Receiver/Transmitter	Full	Point-to-point	Asynchronous
I^2C (Inter-Integrated Circuit)	Half	Bus	Synchronous
SPI (Serial Peripheral Interface)	Full	Bus	Synchronous