Karlsruhe Institute of Technology

Institut fuer Technik der Informationsverarbeitung

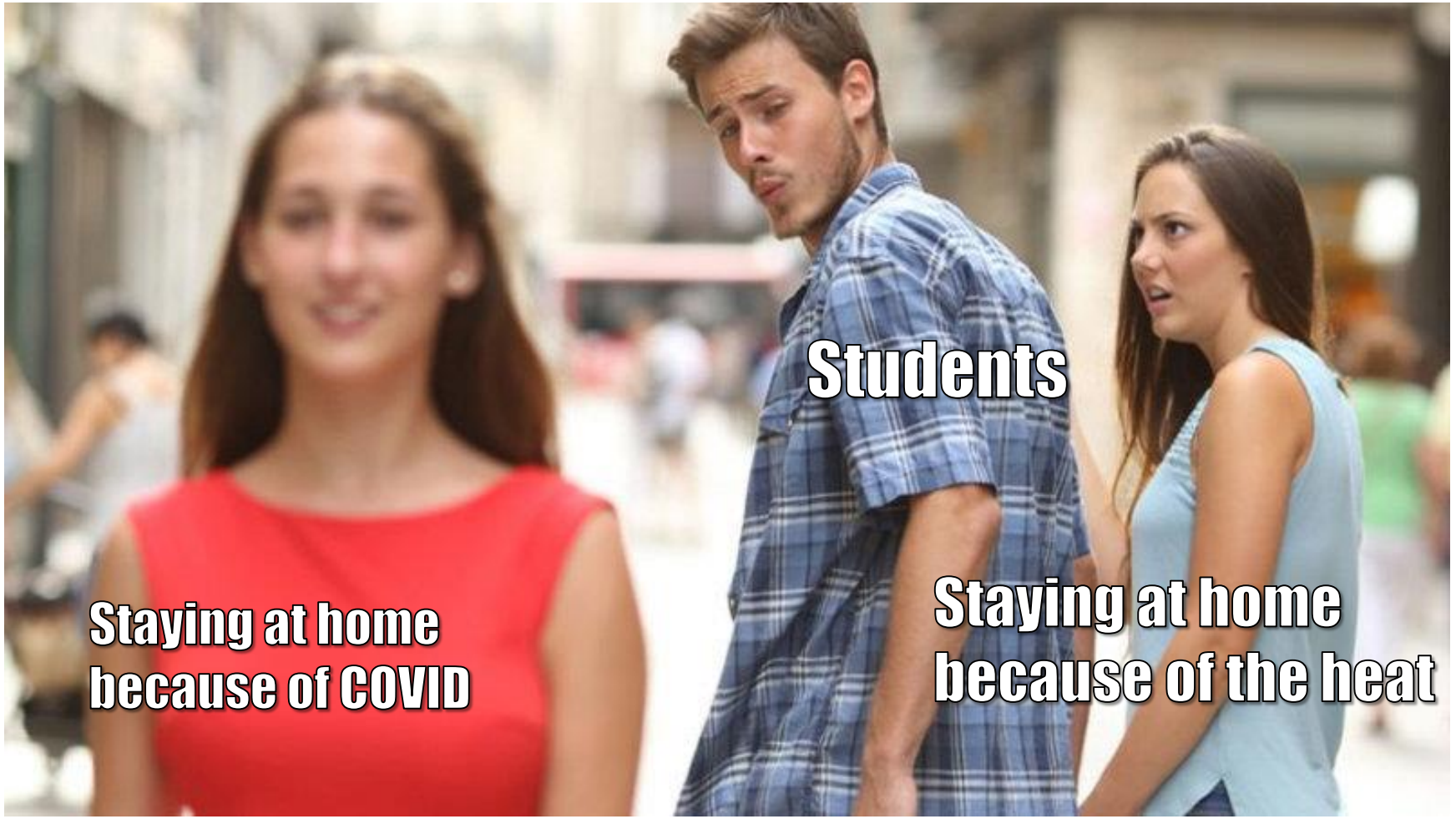
Dr.-Ing. Jens Becker *(jens.becker@kit.edu)* **M. Sc. Matthias Stammler** *(stammler@kit.edu)*

Communication Systems and Protocols

Exercise 5

General Information
Communication Systems and Protocols (Exercises)

Matthias Stammler, M. Sc.
Institut fuer Technik der Informationsverarbeitung (ITIV)
Building 30.10, Room 218
Email: stammler@kit.edu, csp@itiv.kit.edu
Website: https://www.itiv.kit.edu/21_7686.php

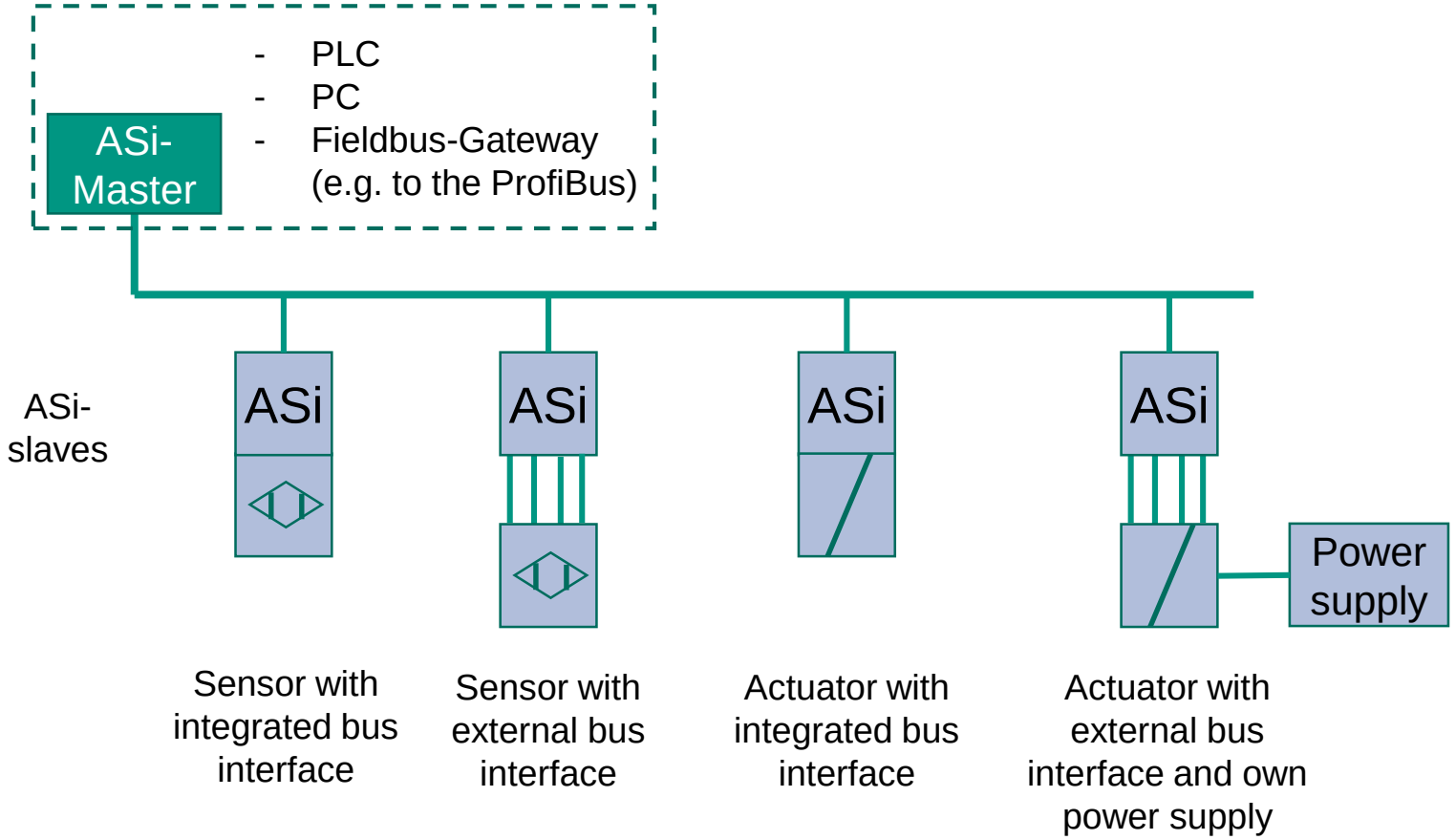

Students
Staying at home
because of COVID
Staying at home
because of the heat

The Actuator Sensor Interface (ASi)

- The goal is to directly connect sensors/actuators to the control via a bus
- Requirements:
  - Easy installation
  - Easy commissioning and maintenance
  - Low-priced
- Data and energy should be transmitted on a two-conductor cable for all sensors and most actuators
- Simple and robust transmission procedure without limitations concerning net topology
- Compact and inexpensive bus connection
- Specified by IEC 62026-2: 2015

Concept of the ASi

- Master slave concept using a single master



Modulation scheme

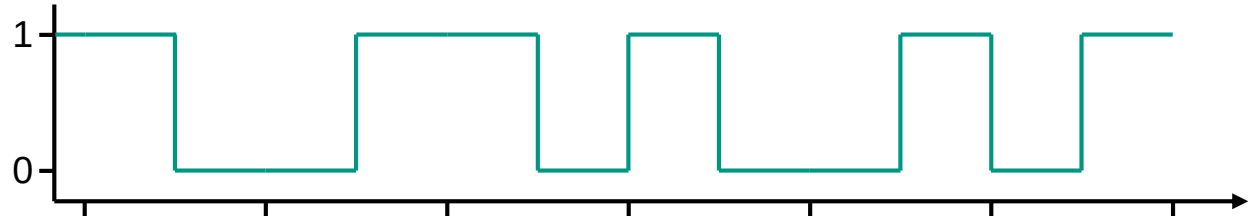
- The modulation scheme must satisfy the following requirements
  - Free of continuous current, since the data signal is modulated on top of the power supply
  - Master and slave must be able to easily generate the signal
  - Narrow-band since the attenuation of the cables rapidly increases with frequency

- Application of the Alternating Pulse Modulation (APM)
  1. Manchester coding of the raw data (→ phase change with every transition of the sender signal
  2. Generation of an appropriate sender current
  3. Sender current induces a voltage level within an inductivity that is existent only once in the system. The induced voltage can be larger than the supply voltage of the sender (→ neg. voltage at current increase, pos. voltage at descent
     - Low cut-off frequency when voltage pulses are formed as $\sin^2$-pulses
     - On specified lines, bit times of 6µs (→167 kBit/s) realizable

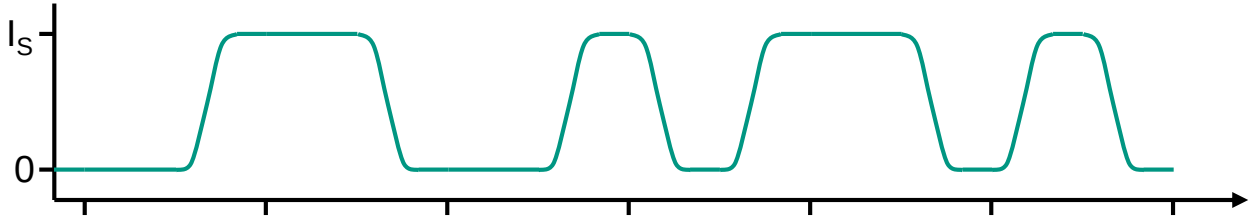
Examples for modulation

Sender bit string

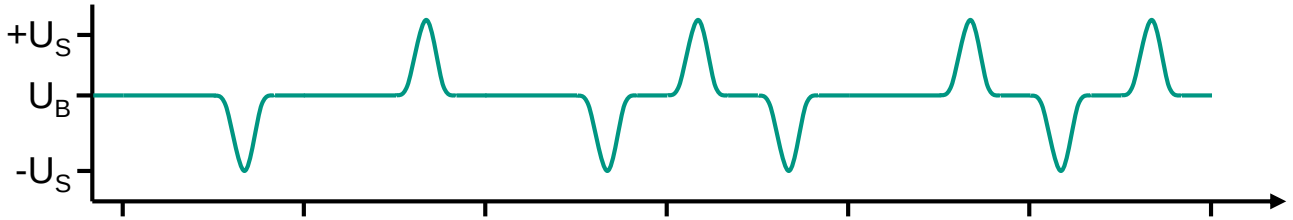
Manchester coded bit string



Sender current



Voltage on signal line

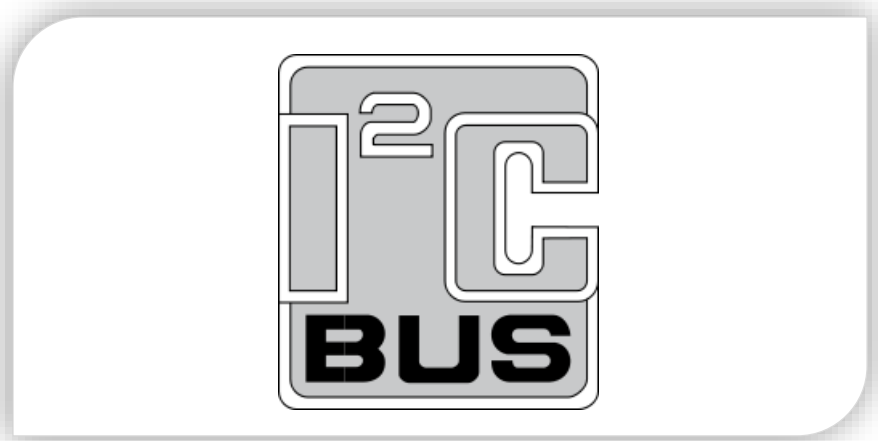


Task 1: ASI

Time Allocated 5 min

I2C-Bus (Inter-Integrated-Circuit-Bus)

- Serial Bus System
- Multi Master / Multi Slave
- CSMA/CR Arbitration Scheme
- Low Cost Bus System



I2C-Bus

- Developed by Philips for interconnecting ICs on printed circuit boards
- Only two wires for data transmission
  - Clock line SCL (serial clock)
  - Data line SDA (serial data)
  - Lower cost and less error-prone because of low pin-count
- Multi-Controller, Multi-Target (Multi-Master, Multi-Slave)
- Serial bus standard used on PCBs or for short distance peripheral IC connection

Source of I²C images and diagrams: *NXP: UM10204, I2C-bus specification and user manual, Rev. 4 — 13 February 2012*

I2C-Bus

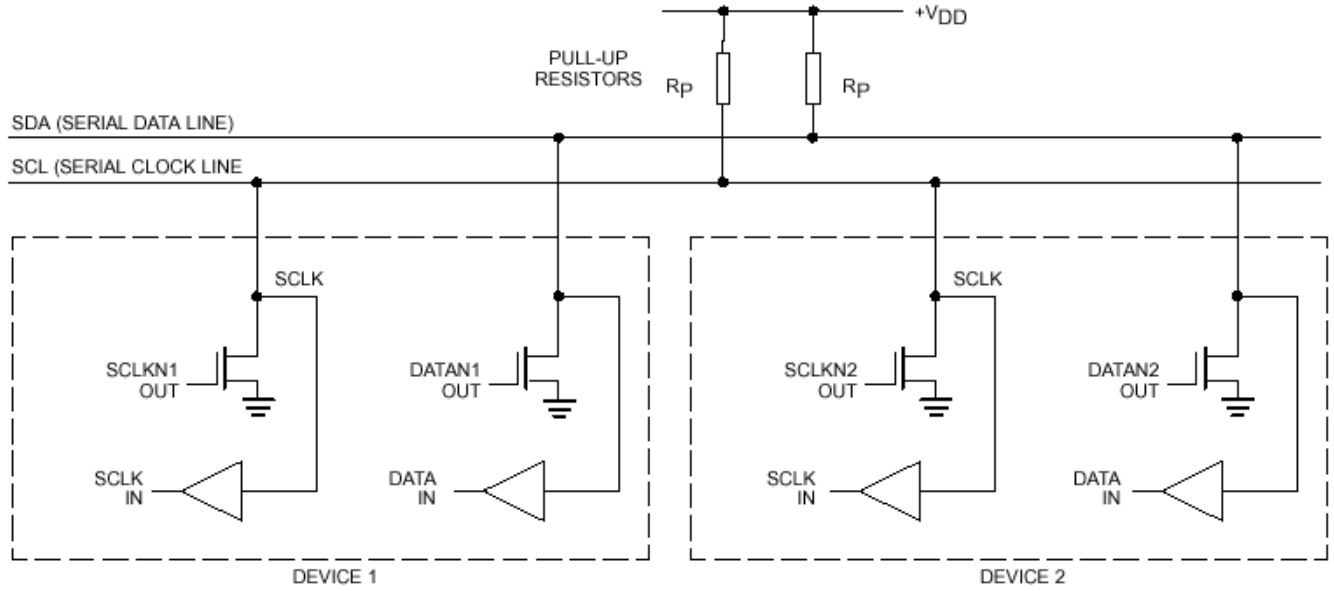
- Data transmission in packets of 8bit
  - 7bit for addresses → 128 addresses
  - 1bit for toggling between reading/writing
- Transfer rate:
  - 100kbit/s in standard mode
  - 400kbit/s in fast mode
  - 3.4Mbit/s in high-speed mode
  - 5.0Mbit/s in ultra fast mode

I2C-Bus

- Each I2C device is recognized by a unique Address.
- Each I2C device can be either a controller (formerly master) or target (formerly slave).
  - Controller (e.g. microcontroller): Initiates a data transfer on the bus.
  - Target (e.g. peripheral): Any device addressed by the Master.
- Half-Duplex transmission after one controller gains bus access

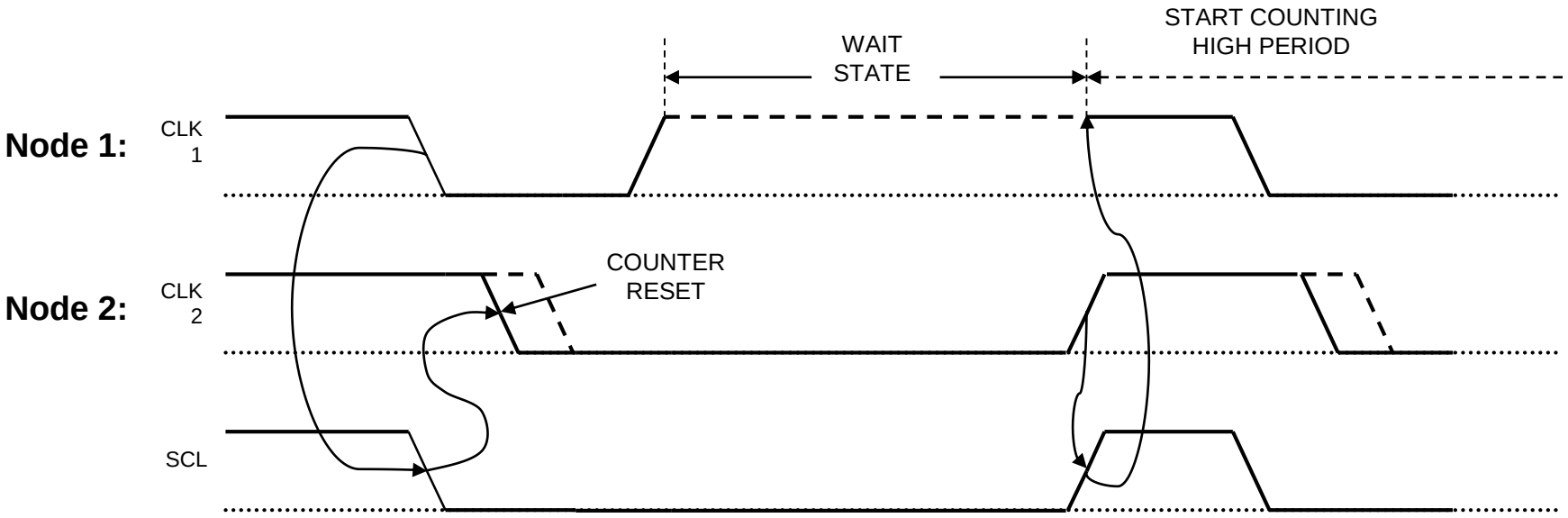
I2C Bus Connection

- Open-Collector outputs for every bus subscriber
  - Wired-AND with dominant ‚LOW‘ on the bus
  - Pull-Up resistors externally connected
- In idle mode, both SCL and SDA are ‚HIGH‘
- Concurrent monitoring of the bus at every subscriber



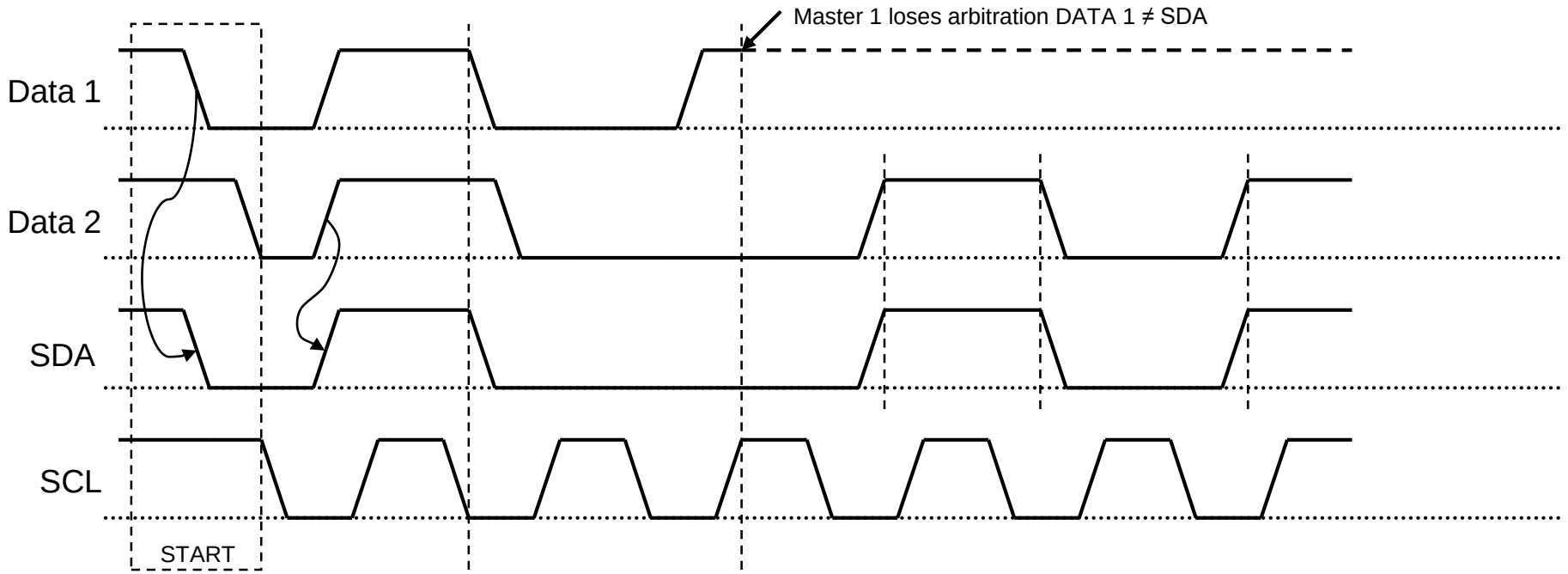
Clock Synchronization

- Clock signal is the sum generated by all nodes
- A slower node can pull down the bus to ‚LOW‘ in order to insert wait states
- The next cycle is not started before SCL is back to ‚HIGH‘



Arbitration

- A master is only allowed to send if the bus is free
  - SCL and SDA are ‚HIGH‘
- Several masters can send at the same time
- While sending, all subscribers read back the bus
- If a master detects that data on the bus is not matching the data it has send, it gets passive and does not send further data for this cycle
- No data loss on the bus → CSMA/CR



Task 2: I²C Synchronization

Time Allocated 10 min

IEEE 1394: Standard for a High-Performance Serial Bus

- Brief History
  - Apple led the initiative in 1987
  - The standard is the result of the collaboration of several companies including Apple, IBM and Sony.
  - Apple called the interface FireWire. It is also known by the brands i.LINK (Sony), and Lynx (Texas Instruments).
  - Approved as IEEE standard in 1995: IEEE 1394-1995
  - Superseded by version in 2008: IEEE 1394-2008
- Version:
  - The description provided here is based on IEEE Std 1394-2008

System Configuration

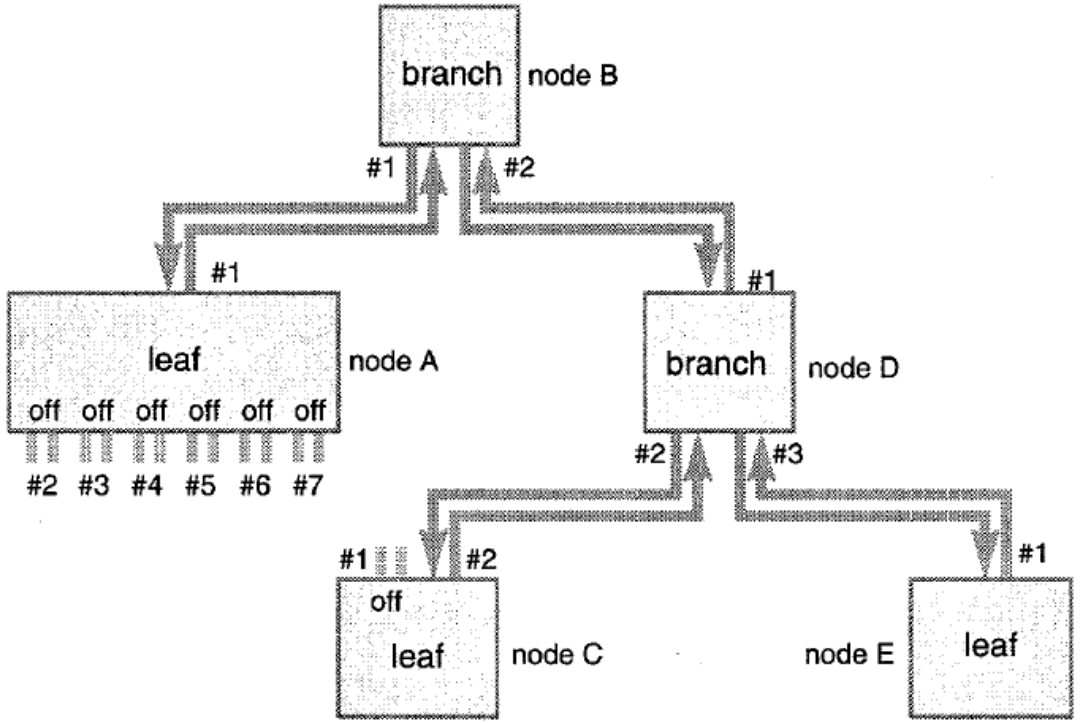
- The cable arbitration takes advantage of the point-to-point nature of the cable environment by having each node handshake with its immediate neighbors to determine ownership of the media

- **Prior to normal operation the system has to be configured**

- System configuration is done in three phases
  1. **Bus initialize**
  2. **Tree identify**
  3. **Self-identify**

1. Bus Initialize

- Whenever a node joins the bus, a bus reset signal forces all nodes into a special state that clears all topology information.
- After the bus initialization process, the only information known to a node is whether it is a
  - **Branch:** more than one directly connected neighbor or
  - **Leaf:** only a single neighbor or
  - **Isolated:** unconnected
- The eventual root may be either a branch or a leaf!



After Bus Initialization

2. Tree identify process

- The tree identify process translates the general network topology into a tree, where one node is designated a root and all of the physical connections have a direction associated with them pointing towards the root node.

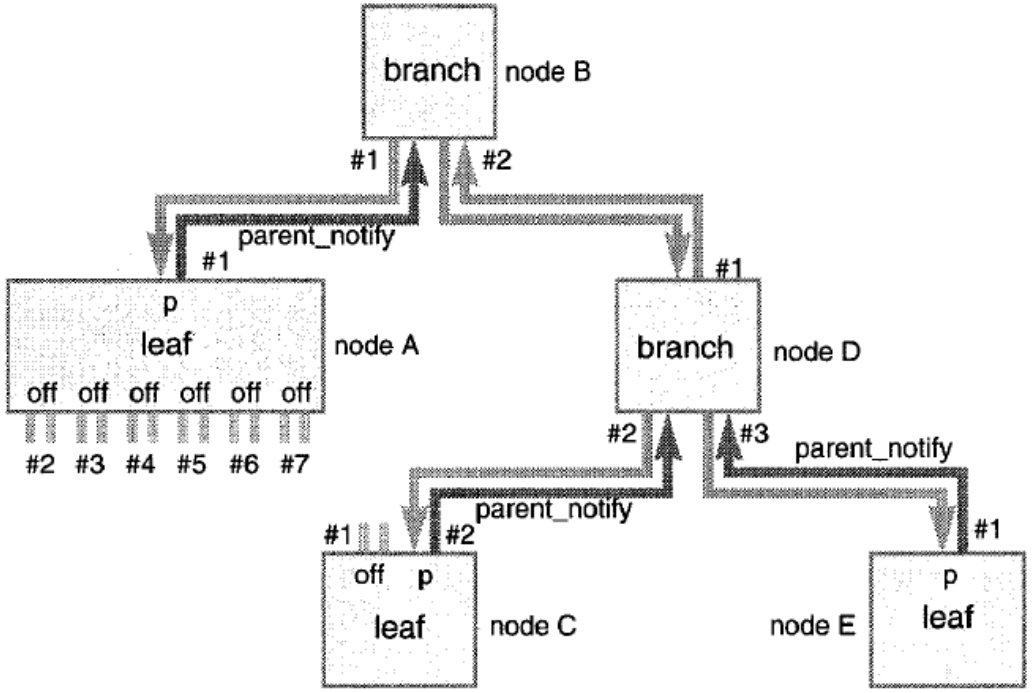
- The direction is set by labeling each connected port as a
  - **Parent:** connected to a node closer to the root or
  - **Child:** port connected to a node further from the root

- Any unconnected ports are labeled “off” and do not participate in further arbitration processes.

- Any loop in the topology is detected by a timeout in this process.

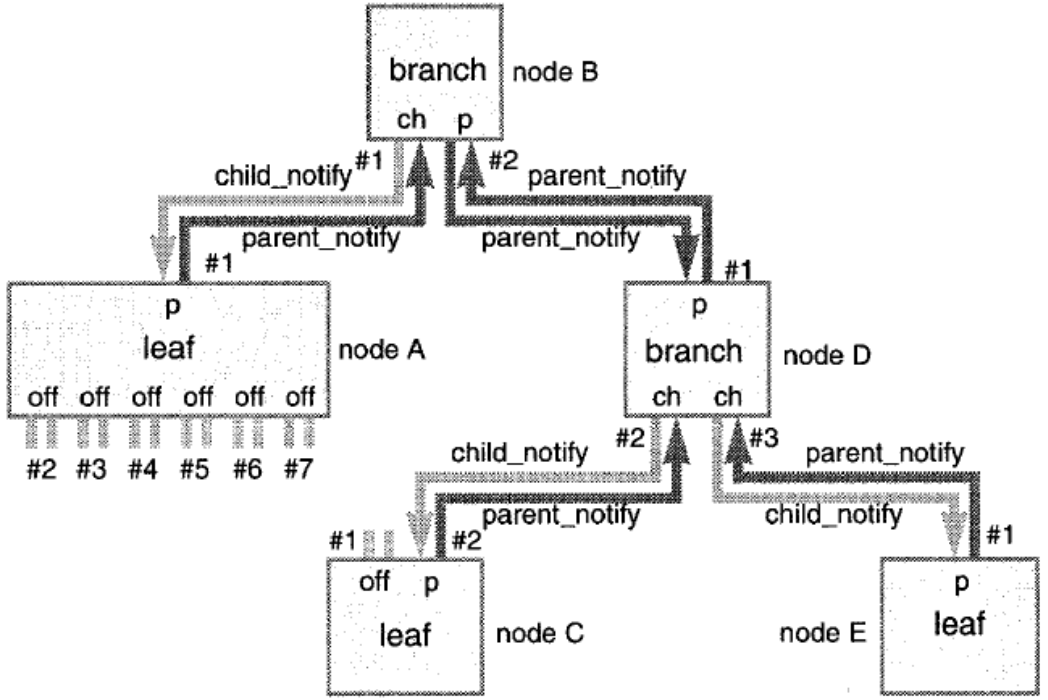
2. Tree identify process

- The first step is for all leaf nodes to notify their probable parents. This is done by sending a *parent_notify* packet (signal)
- In this example, nodes A, C, and E send a *parent_notify* to their single connected port.
- This is the start of the parent-child handshake process.



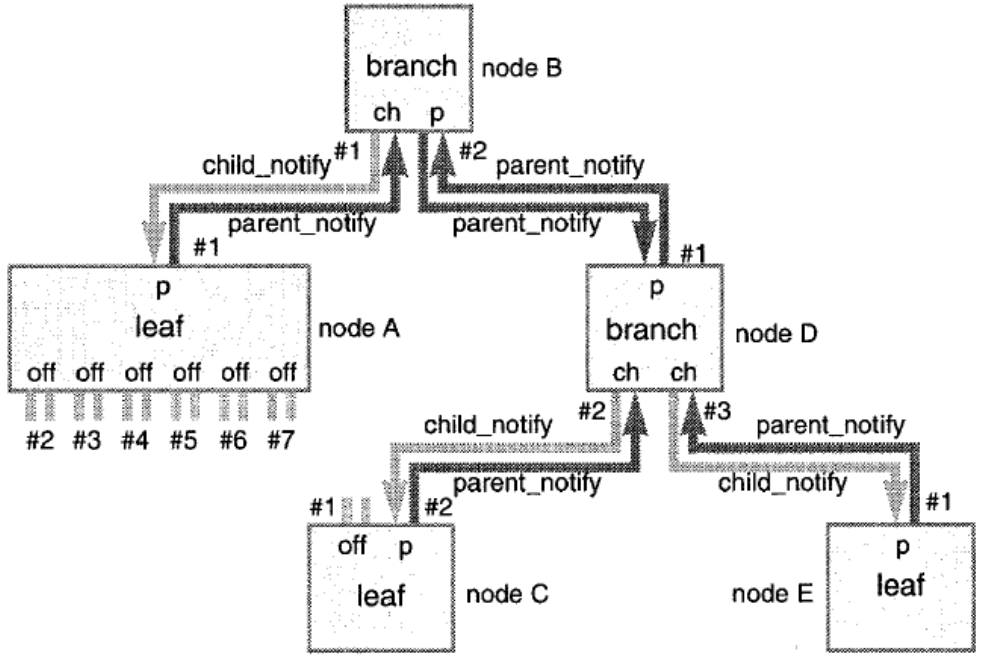
2. Tree identify process

- The nodes internally recognize the *parent_notify* signals and mark the ports receiving them as child ports.
- If these nodes have one unidentified port, they send *parent_notify* up to their probable parents.
- At the same time the nodes send down *child_notify* signals to their child ports.
- Example here: Nodes B and D have one port remaining that is connected but not yet identified as child or parent. Therefore a *parent_notify* signal is sent.



2. Tree identify process

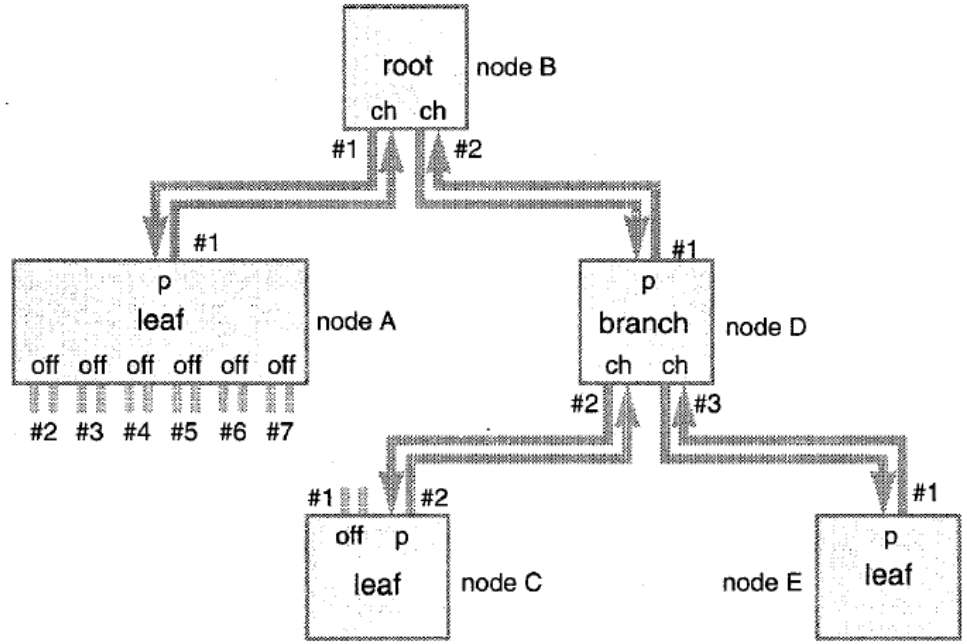
- When the leaf nodes receive the *child_notify* signal, that port is assigned as their parent port. Once assigned, their *parent_notify* signal is withdrawn.
- **Nodes having only child ports are assigned as the root.**
- It can occur that multiple nodes are receiving only *parent_notify* signals. This leads to a process called “root contention”
- Example:
  - Both nodes B and D discover that they are receiving the *parent_notify* port
  - Since one of the two nodes has to become the parent of the other, this collision of intentions starts a process called “root contention”.
- This is resolved by withdrawing the *parent_notify* signals. A timer is assigned on each node with a random duration and the signals are resent after the time elapses.



Start of root contention

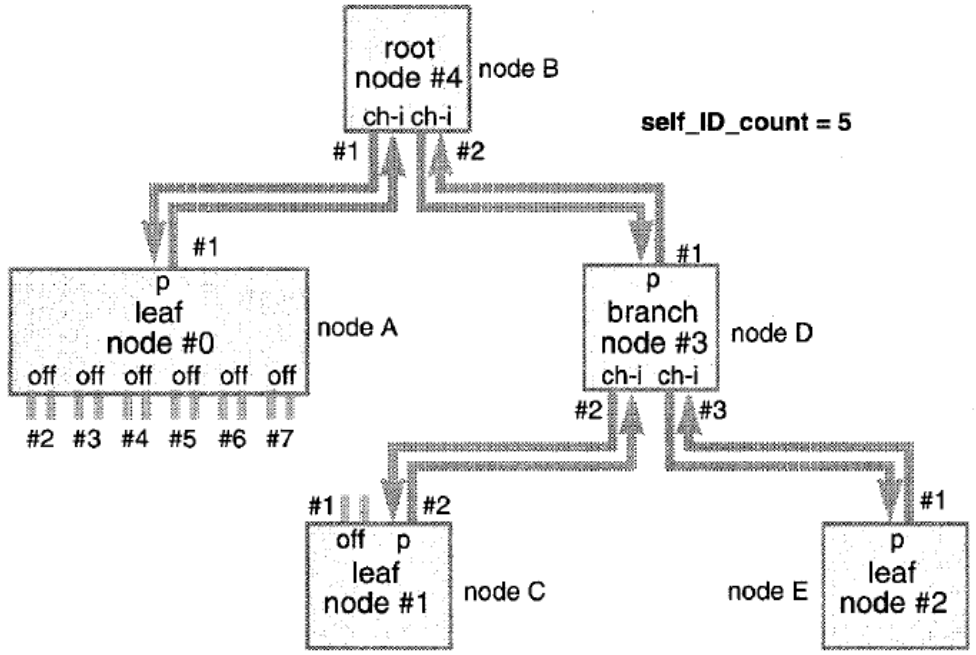
2. Tree identify process

- When a node has all ports labeled as children, it takes the root function for itself.
- In this example:
  - Node D resends the signals first due to root contention
  - Node B has all ports as children and is assigned as the root.
- Note that the selection of the root node is not topology dependent. It is completely acceptable that the root node also be a leaf.
- The node that waits the longest after the bus reset to start participating in the tree identify process becomes the root.
- A particular node can be forced to wait a longer time by setting a *force_root* bit



3. Self-Identify Process

- The next step is to give each node a unique physical_ID
- The self-identify process uses a deterministic selection process, where the root node passes control of the media to the node attached to its lowest numbered connected port first. The child nodes then passes control in a recursive manner
- The first leaf node to receive control, during its self-identify process selects 0. The next node which receives control selects 1 and so on
- The root then passes control to the next lowest numbered port. When the nodes attached to all the ports of the root are finished, the root itself does a self-identify process.
- **Note that each port of the node is individually numbered. There is no particular order to the numbering, it is just a way to give each port a unique label**



Task 3: Fire Wire

Time Allocated 10 min

Thank you for your attention