Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung ITIV

Dr.-Ing. Jens Becker (jens.becker@kit.edu) ***M.Sc. Nidhi Anantharajaiah*** (nidhi@kit.edu)

Institut für Technik der Informationsverarbeitung (ITIV)

Communication Systems and Protocols

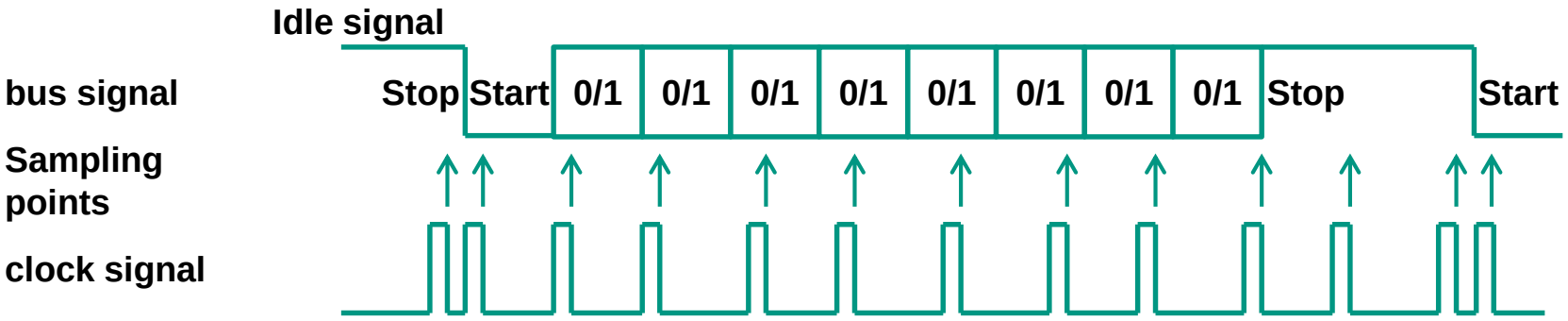
Exercise 4

Synchronisation Schemes

	Synchronous Transmission	Asynchronous Transmission
Parallel Transmission	Shared clock line	Flow-Control
Serial Transmission	Suitable line code or scrambler (shared clock line)	Start-Stop-mode

Start-Stop mode

- When no transmission occurs, bus is in idle state → no signal changes occur on the bus
- Begin of a transmission is signaled using a well defined edge on the bus signal → start bit
- This edge is used to synchronize sender and receiver clock
- Every bit has a well defined step size and thus can be separated using the internal clocks → baud rate
- At the end, it has to be ensured that the bus goes to idle level in order to be able to detect a new transmission → stop bit

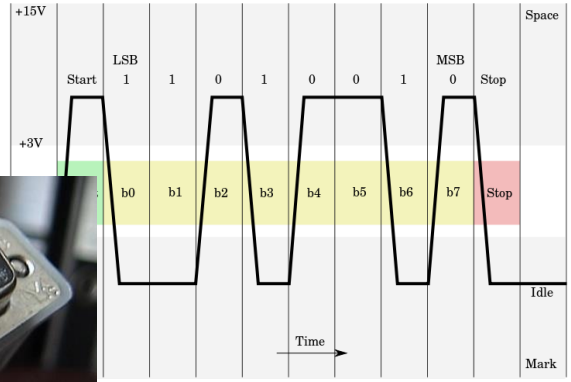


Start-Stop mode: Format definition

- Format definition for serial transmission using start-stop mode
  - **X-Y-Z**
- With:
  - X: number of data bits (normaly 5-8)
  - Y: parity check (none, even, odd)
  - Z: number of stop bits (1, 2)
- Notes:
  - There is always on start bit (denotes begin of transmission)
  - The number of stop bits in the definition is the **minimum** number. There can be more „stop bits“ if no transmission is going on

Recommended Standard 232 RS-232

- Standard for serial communication transmission of data
  - used for connections to modems, printers, data storage, …
- RS-232 was first introduced in 1962
  - by the *Radio Sector* of the EIA (Electronic Industries Alliance)
- USB has displaced RS-232 from most of its peripheral interface roles
  - still used in areas of micro controllers and programming



Source: https://de.wikipedia.org/wiki/RS-232

Task 1: Serial Interface

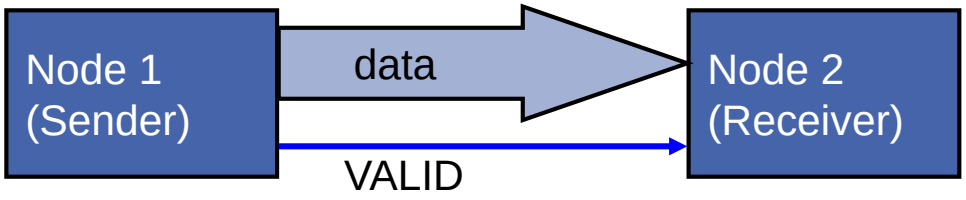
Time Allocated 10 min

Synchronisation Schemes

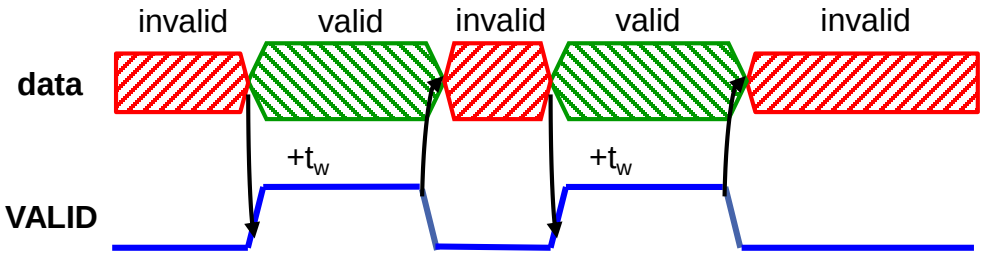
	Synchronous Transmission	Asynchronous Transmission
Parallel Transmission	Shared clock line	Flow-Control
Serial Transmission	Suitable line code or scrambler (shared clock line)	Start-Stop-mode

Open-loop Flow Control

If data is transmitted in multiple subsequent cycles, it has to be regulated when data of one cycle is processed and when a new cycle may be started.



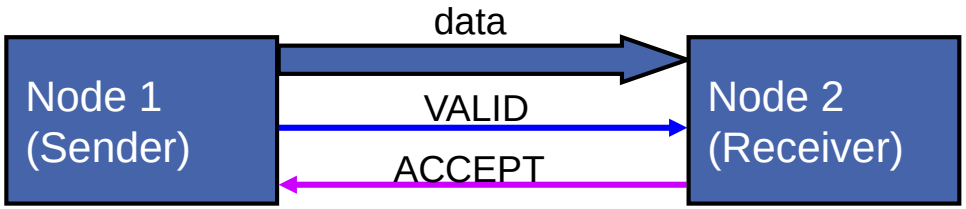
- One always waits for a fixed period of time t_w



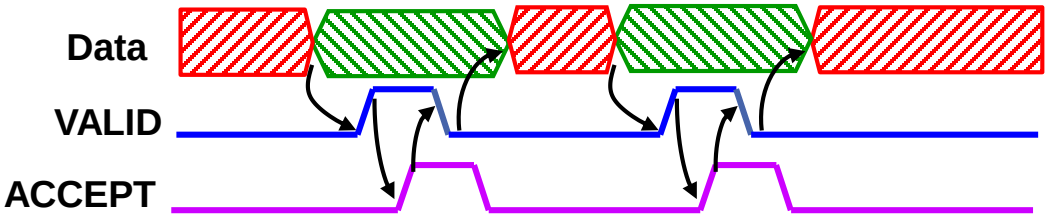
  - Disadvantage: Waiting time t_w is dependent on the slowest node on the bus and is always invariant in length (synchronous mode)

Level-triggered Closed-loop Flow Control

If data is transmitted in multiple subsequent cycles, it has to be regulated when data of one cycle is processed and when a new cycle may be started.



- Receiver asserts ACCEPT signal if data is not needed any longer.

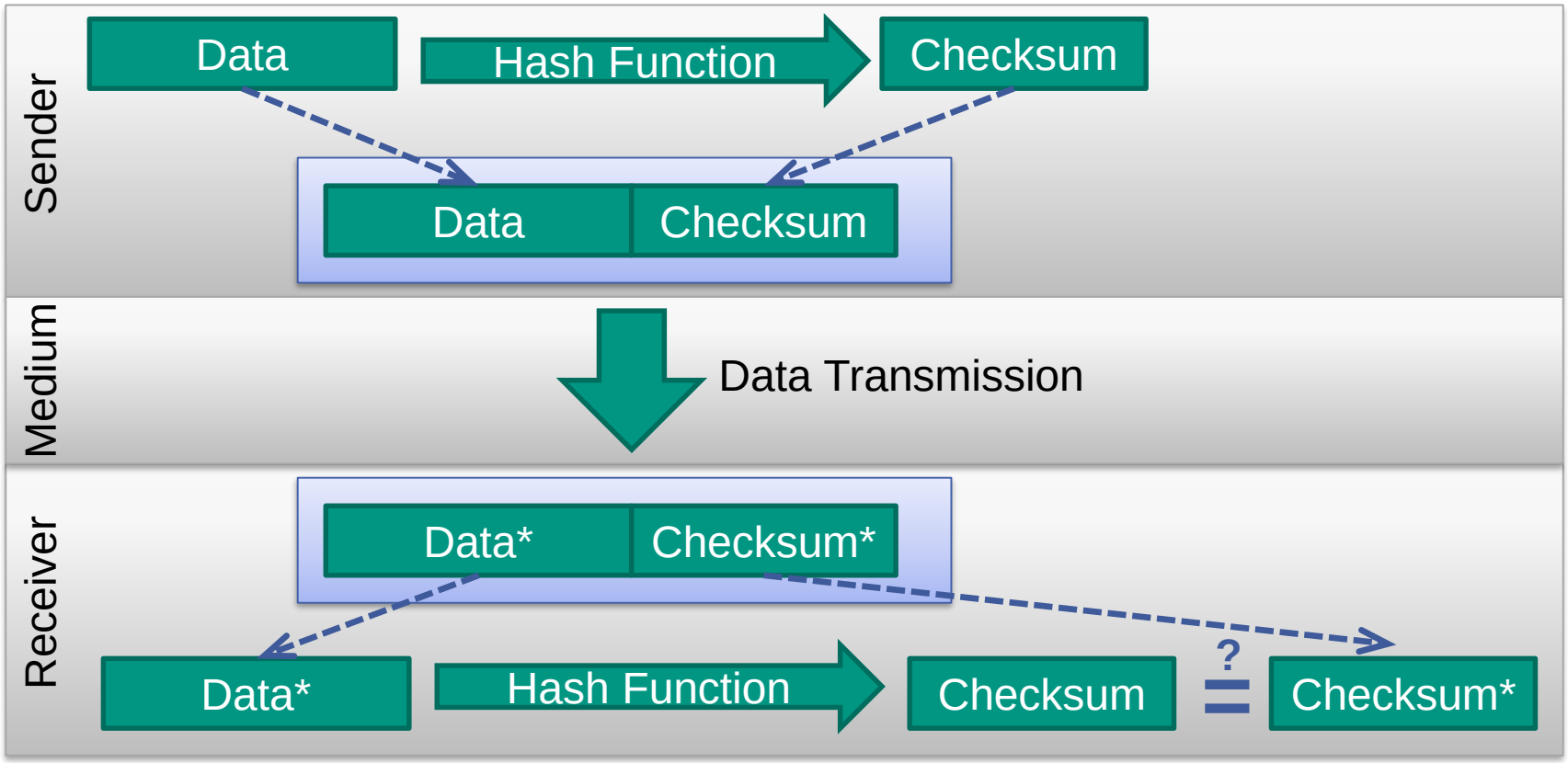


Task 2: Flow-Control

Time Allocated 5 min

Cyclic Redundancy Check (CRC) – Approach

- Generate a checksum (Hash)
- Send checksum together with raw data
- Receiver checks received data and received checksum match

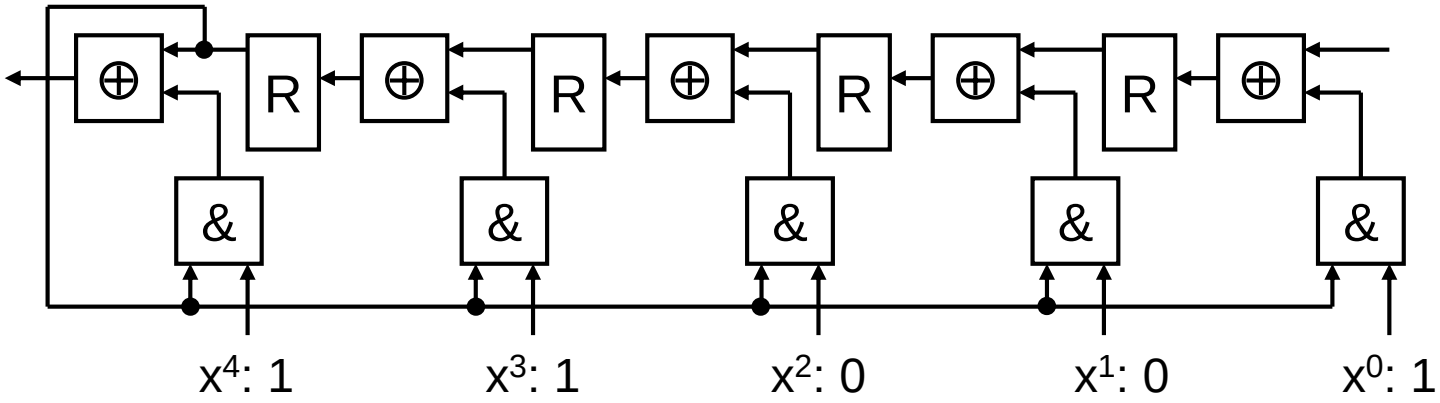


CRC –mathematical description

- Generation of checksum uses a generator polynomial *G(x)* on sender and receiver side
  - most significant bit (MSB) and least significant bit (LSB) must be ‚1‘
- Calculation on sender side
  - Data vector *M(x)* of length *m* bit (exceeding length of *G(x)*)
  - Checksum equals the modulo of the division $(x^r M(x))/G(x)$
    - *r:* degree of generator polynomial
    - $x^r M(x)$ adds *r* zero bits to the end of the data vector
  - Checksum is appended to the data
    - Corresponds to the addition of the modulo: $x^r M(x) + R$
- Calculation on receiver side
  - Division of received data by *G(x)*
    - If the modulo of the division is not equaling zero an error has occurred
    - If the modulo of the division equals zero, no error has been detected

Hardware Implementation of CRC

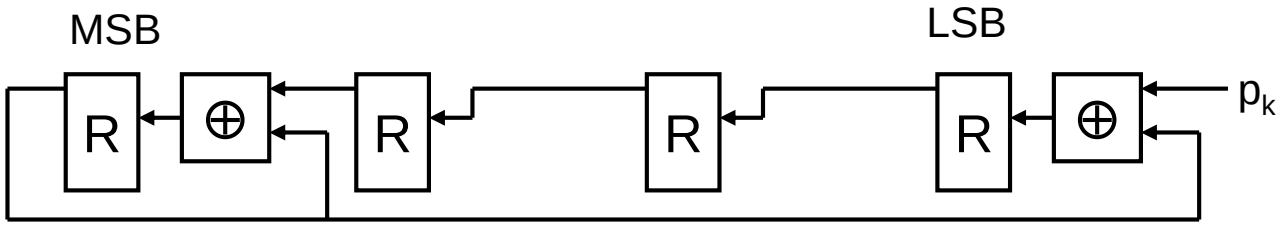
- Utilization of linear feedback shift registers with XOR-operations
- Example: Generator polynomial **$x^4+x^3+1 = 11001$**



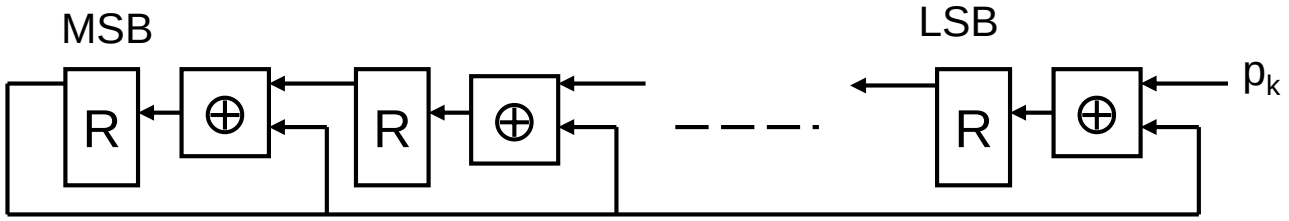
- Data vector is shifted bitwise through the register
- Content of MSB denotes the current quotient bit
- If MSB=1, the generator polynomial is added (XOR gate) and the dividend is shifted left (corresponds to a right-shift of the generator)
- If MSB=0, only the shift is executed
- Input data is shifted until a logic 1 is available in the MSB

Hardware Implementation

- Simplification for fixed generator polynomial:
- Example: Generator polynomial **$x^4+x^3+1 = 11001$**
  - AND-Gates can be omitted
  - XOR-gates only required for factors $\neq 0$ in the generator polynomial
  - last XOR-gate can be omitted as the output is not used



- General representation



Task 3: CRC

Time Allocated 10 min