CprE 488 – Embedded Systems Design

Lecture 4 – Interfacing Technologies

Joseph Zambreno
Electrical and Computer Engineering
Iowa State University

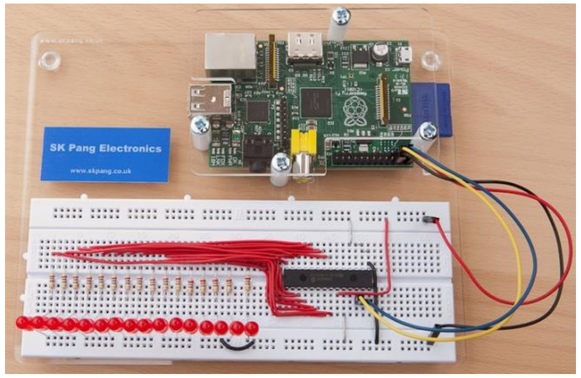
www.ece.iastate.edu/~zambreno
rcl.ece.iastate.edu

Never trust a computer you can't throw out a window. – Steve Wozniak

System Connectivity

- The communication strategies we have explored so far (direct memory mapped I/O, shared buses and memory) are most commonly found in SoCs
- For multi-chip / multi-board systems, the communication needs can be different:
  - Performance is not (necessarily) a significant factor
  - Simplicity in terms of hardware and wiring (cost)
  - Configurability, expandability
- Many one-off solutions, but several formal and informal standards have emerged as well

Freescale Tower System
Modular Development Platform



Raspberry Pi with i2c
I/O Expander

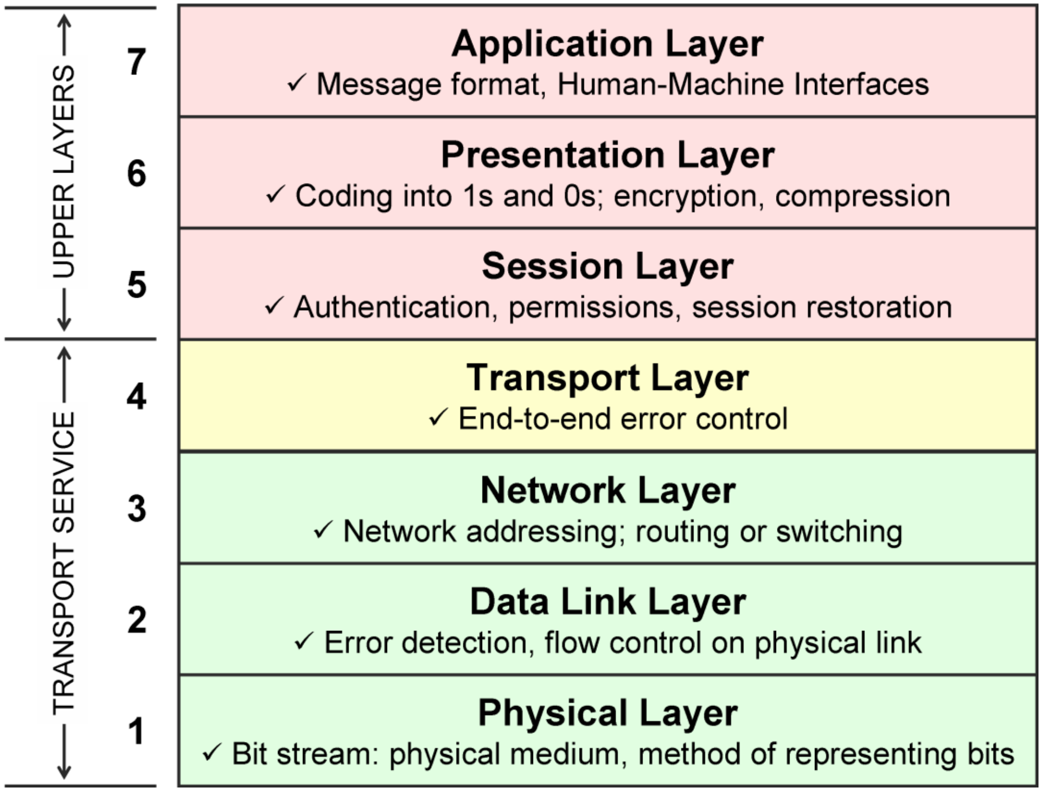
This Week's Topic

- Serial communication bus standards
  - Inter-Integrated Circuit (I^2C)
  - Serial Peripheral Interface (SPI)
  - Controller Area Network (CAN)
  - Quick Path Interconnect (QPI)

- Reading: Wolf chapter 9.3, 10.4.5

A Note on Network Stacks

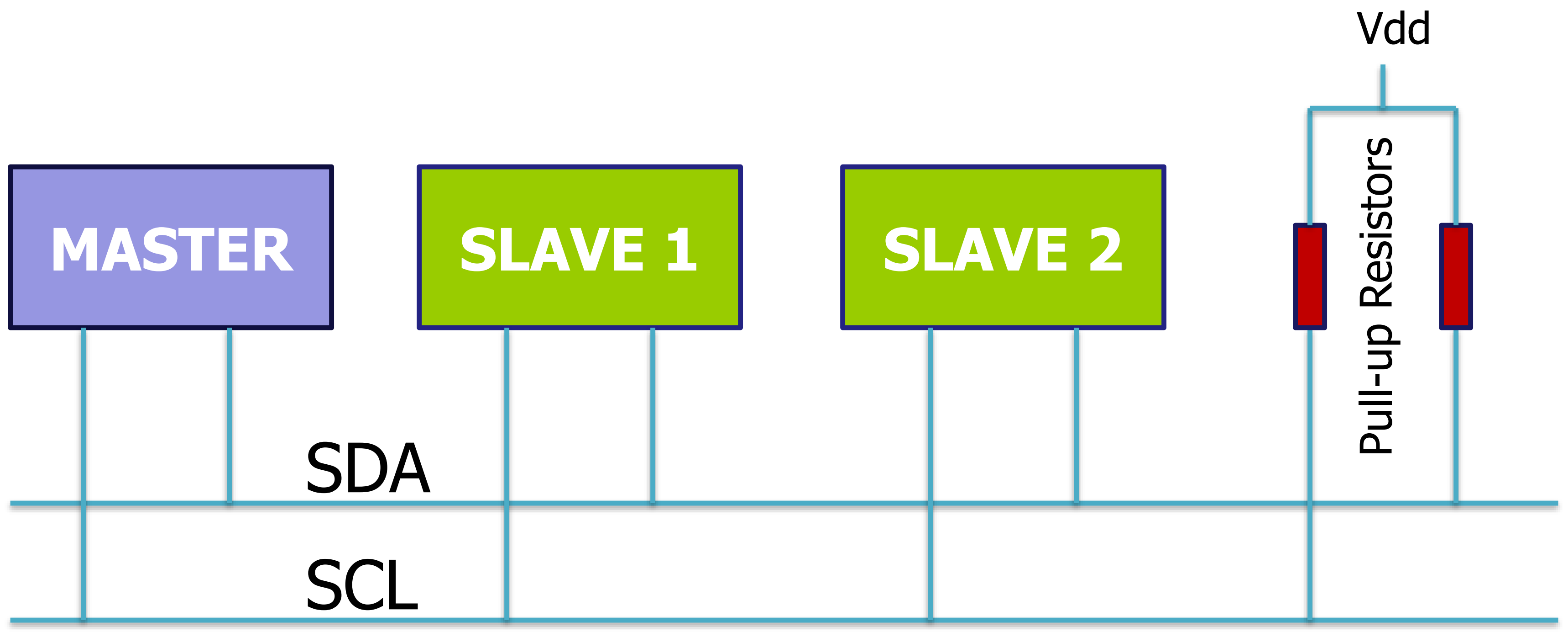
- Networks (including the serial buses we are discussing) can be classified using the well-known ISO-OSI model:


UPPER LAYERS
TRANSPORT SERVICE
7
Application Layer
✓ Message format, Human-Machine Interfaces
6
Presentation Layer
✓ Coding into 1s and 0s; encryption, compression
5
Session Layer
✓ Authentication, permissions, session restoration
4
Transport Layer
✓ End-to-end error control
3
Network Layer
✓ Network addressing; routing or switching
2
Data Link Layer
✓ Error detection, flow control on physical link
1
Physical Layer
✓ Bit stream: physical medium, method of representing bits

- Our interest is mainly in layers 1 and 2, with software (potentially) providing higher level services

Inter-Integrated Circuit (I^{2}C) Bus

- Low-bandwidth (100s of Kbps), short-distance, two-wire interface for communication amongst ICs and peripherals
- Originally developed by Philips Semiconductor for TV circuits, later became an industry standard

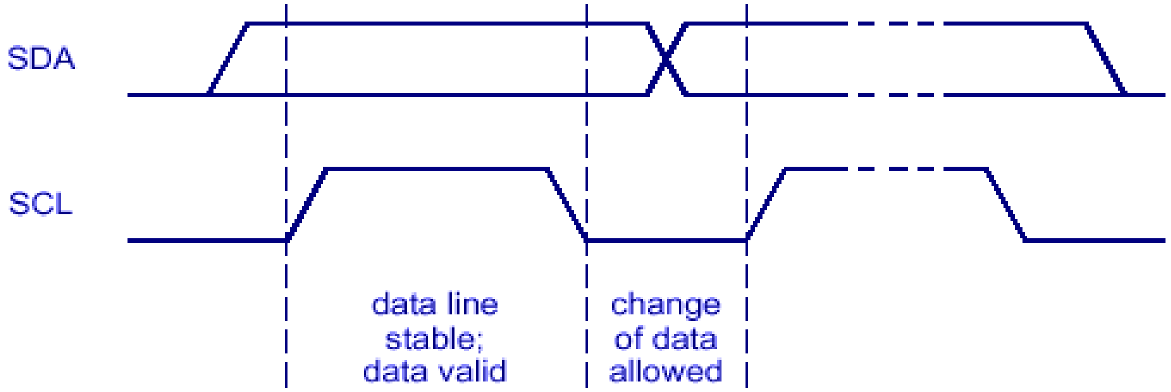


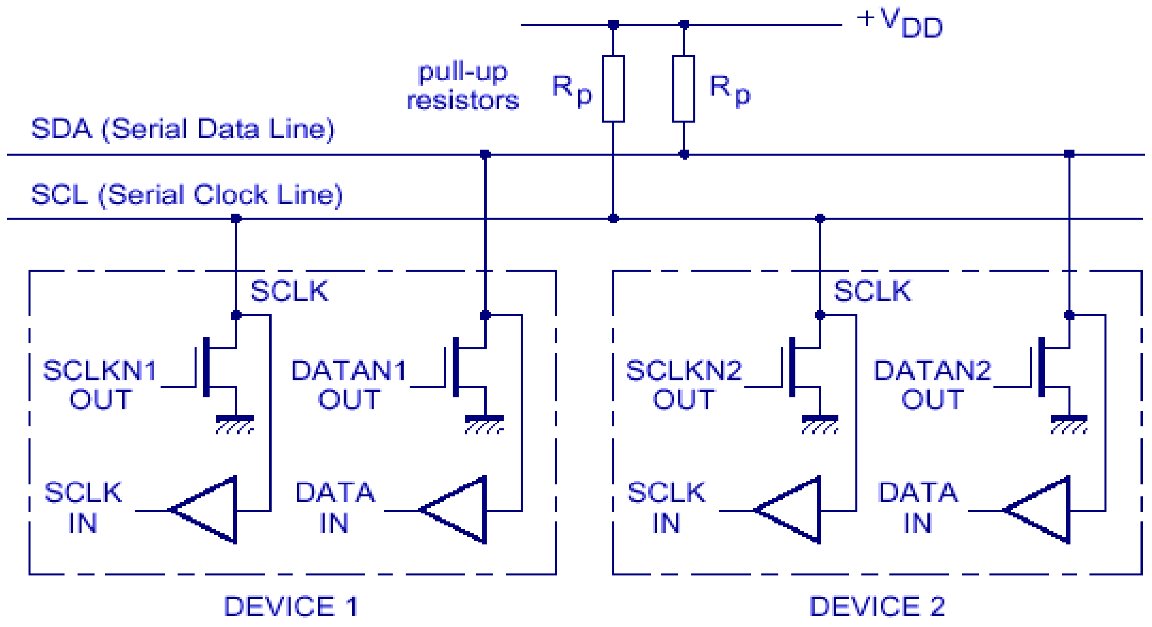
I^{2}C Features

- Multiple receivers do not require separate select lines as in other buses
  - At start of each I^{2}C transaction a 7-bit (or 10-bit) device address is sent
  - Each device listens – if device address matches internal address, then device responds
- SDA (data line) is bidirectional, communication is half duplex
- SDA, SCL are open-drain, require external pullup resistors
  - Allows multiple bus masters

I^{2}C Bit-Transfer

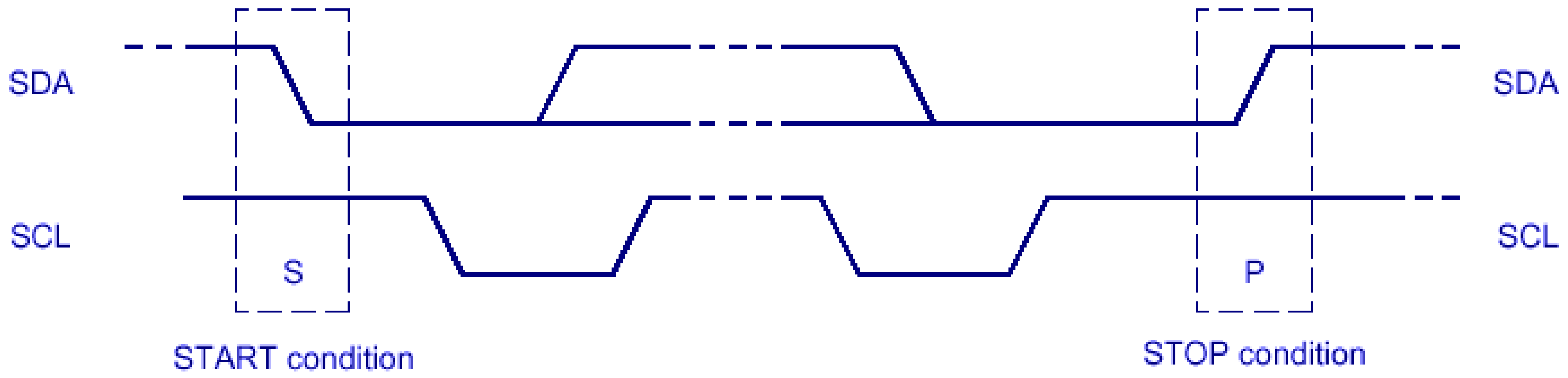
- One clock pulse is generated for each data bit that is transferred
- Data Validity
  - The data on the SDA line must be stable during the high (1) period of the clock
  - SDA can change data only when the SCL is low (0)
- Wired-and function
  - open-drain or open-collector





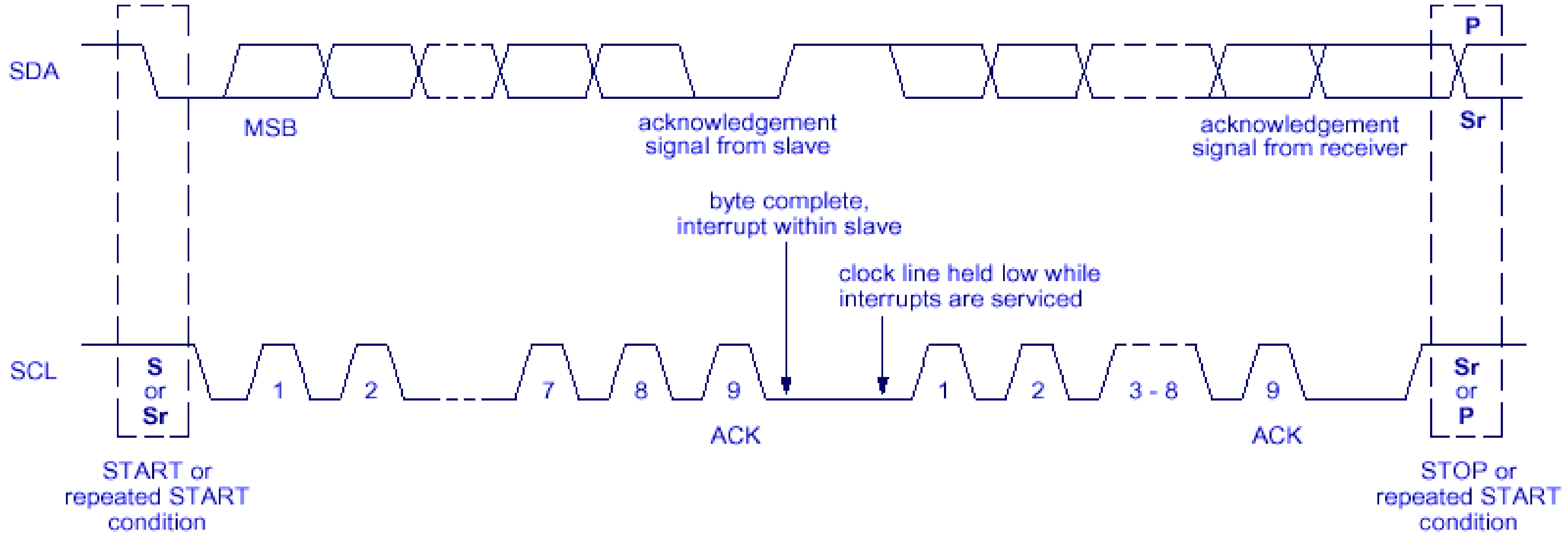
I^{2}C START/STOP Conditions

- START condition: Signals begin of transfer (occupies the bus)
  - A HIGH to LOW transition on the SDA line while the SCL is HIGH
- STOP condition: Signals end of transfer (releases the bus)
  - A LOW to HIGH transition on the SDA line while the SCL is HIGH
- Both these are always generated by the Master
- Repeated START condition is allowed
  - Repeated start is used for changing the slave, or changing the direction of data transfer (Send/Receive) for the same slave



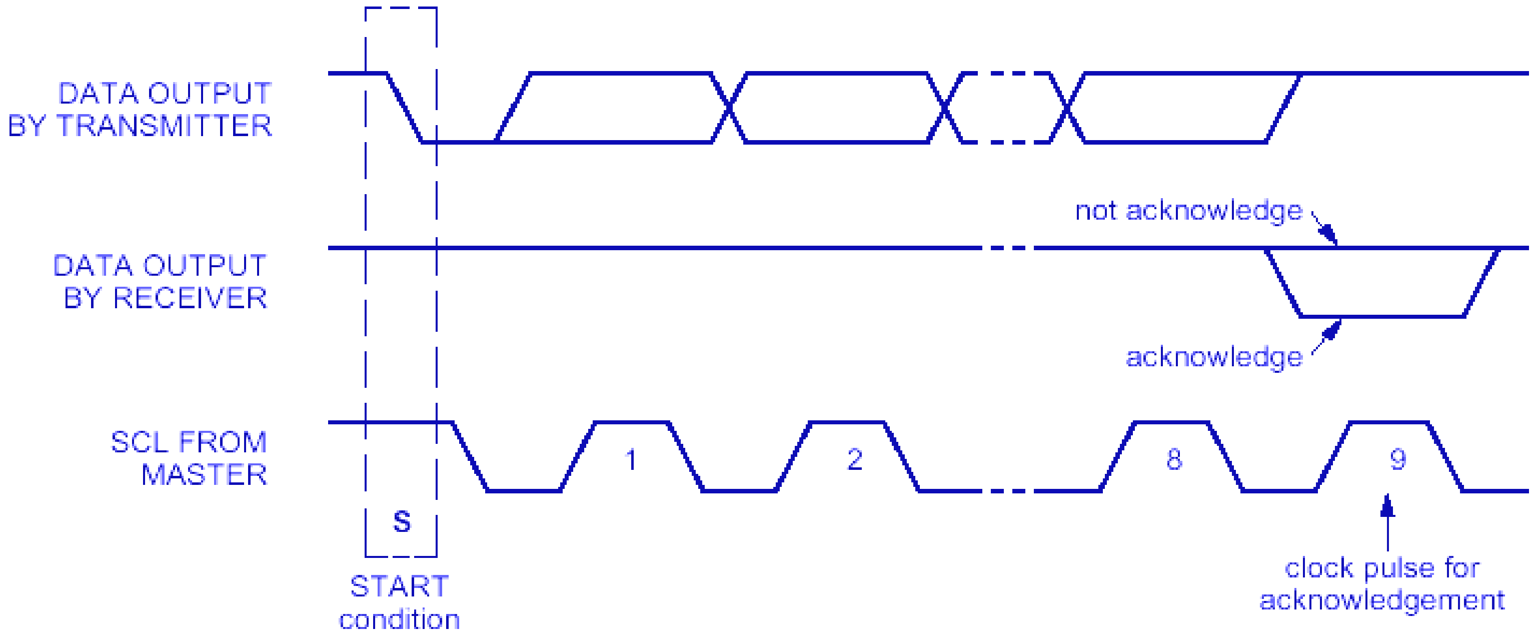
I^{2}C Data Transfer

- Every byte on the SDA line must be 8-bits long
- Each byte must be followed by an acknowledgement from the receiver
- Data byte is transferred bit-wise with the MSB as the first bit sent
- A slave can force the master to wait by holding the clock line SCL LOW


SDA
SCL
MSB
acknowledgement signal from slave
acknowledgement signal from receiver
byte complete, interrupt within slave
clock line held low while interrupts are serviced
S or Sr
1
2
7
8
9
ACK
1
2
3 - 8
9
ACK
P
Sr
Sr or P
START or repeated START condition
STOP or repeated START condition

Acknowledgement Scheme

- The acknowledge-related clock-pulse is generated by the master
- The transmitter (master or slave) releases the SDA line i.e. SDA is HIGH for the ACK clock pulse
- The receiver must pull-down the SDA line during the acknowledge clock pulse (stable LOW) during the HIGH period of the clock pulse

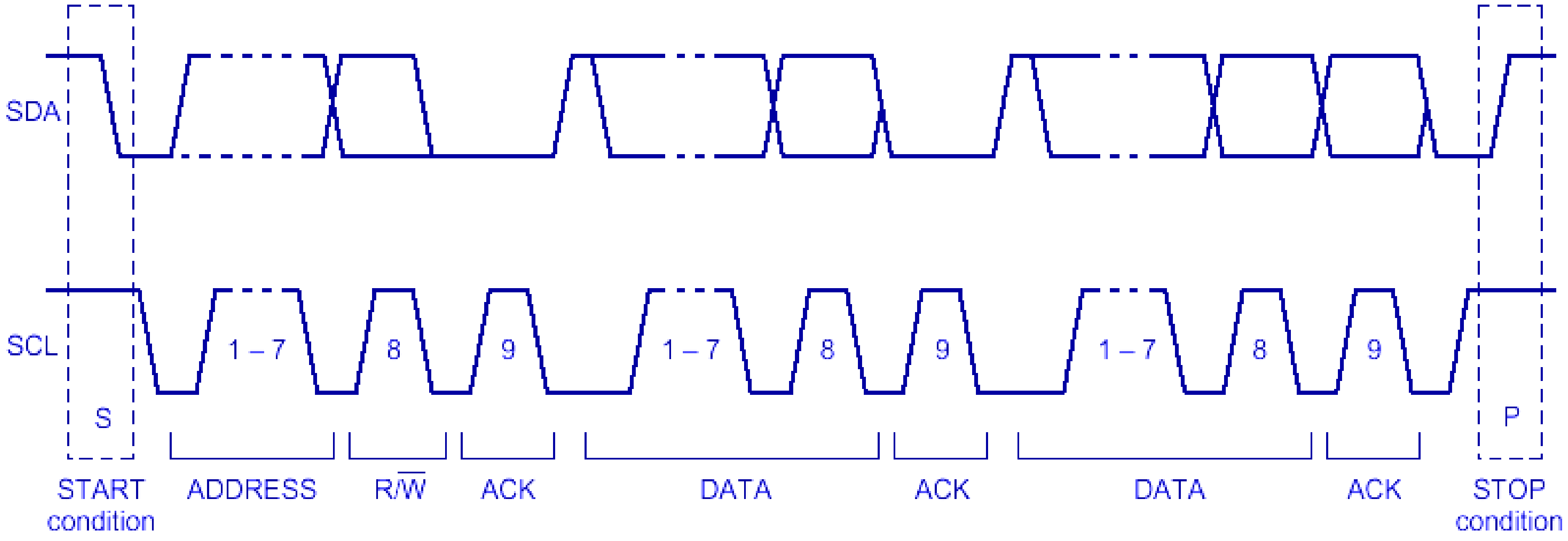


Acknowledgement Scheme

- The receiver is obliged to generate an acknowledge after each byte received
- When a slave does not acknowledge slave address (when busy), it leaves the data line HIGH. The master then generates either STOP or attempts repeated START
- If a slave-receiver does ack the slave address, but some time later during the transfer cannot receive more data (this is done by leaving SDA HIGH during the ack pulse), then the master either generates STOP or attempts repeated START
- If a master-receiver is involved in a transfer, it must signal the end of data to the slave-transmitter by not generating an ack on the last byte that was clocked out of the slave. The slave-transmitter must release the data line to allow the master to generate a STOP or repeated START condition

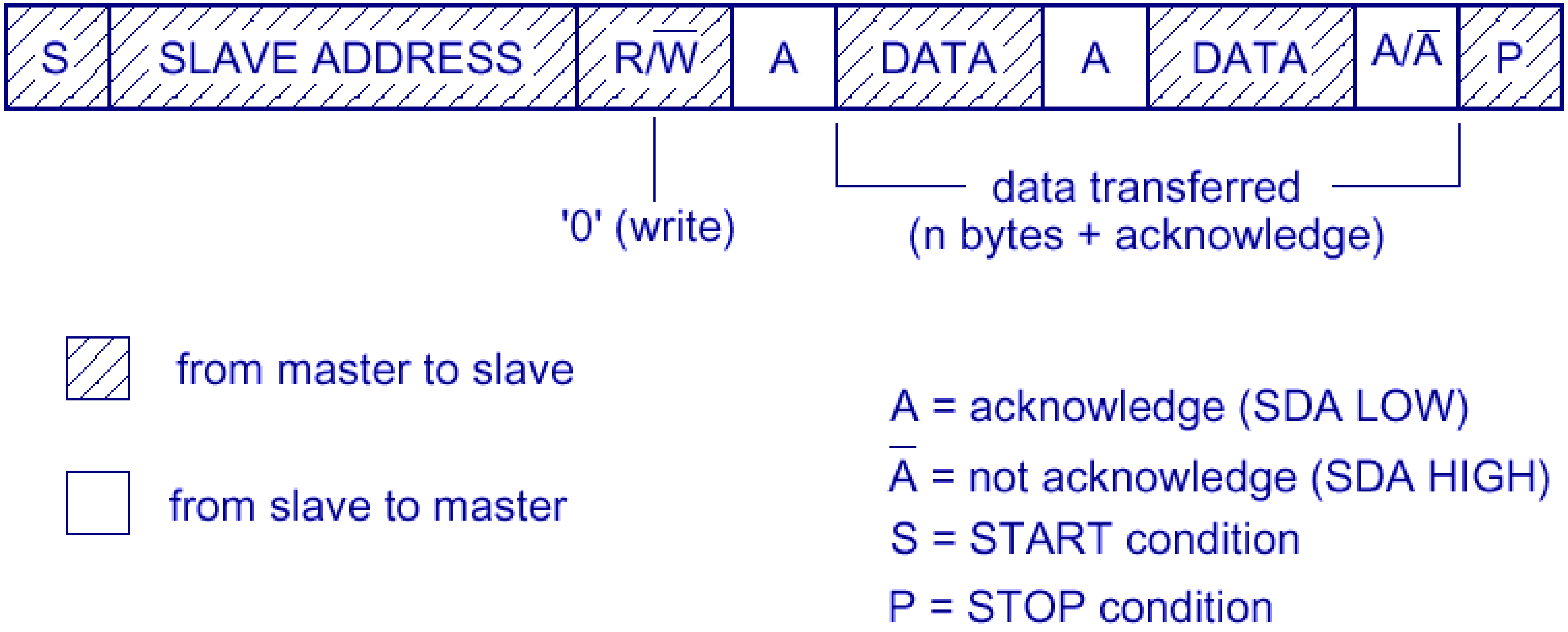
Data Transfer With 7-Bit Device Address

- After START condition (S), a slave address (7-bit) is sent.
- A read/write (R/W’) direction is then sent (8th bit)
- Data transfer occurs, and then always terminated by STOP condition. However, repeated START conditions can occur.



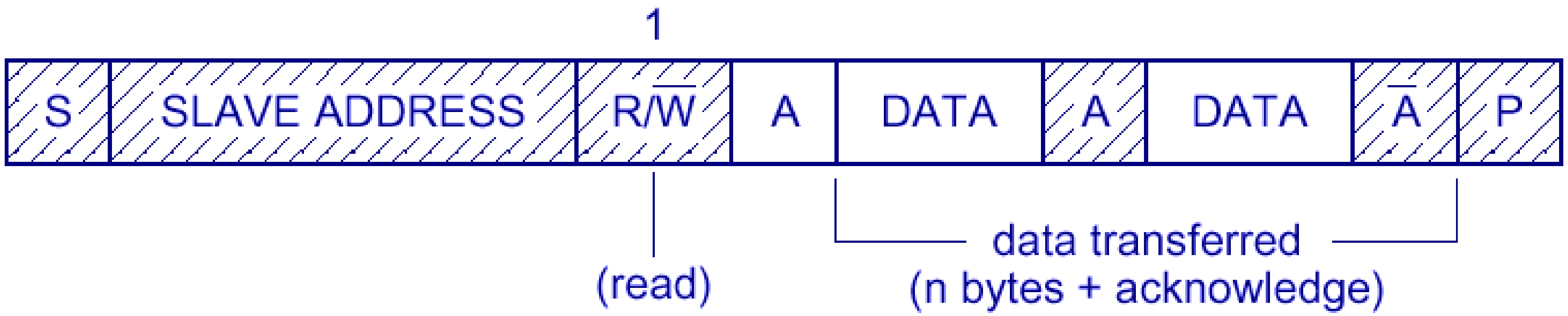
Master to Slave Data Transfer

- In this configuration, the transmission direction never changes



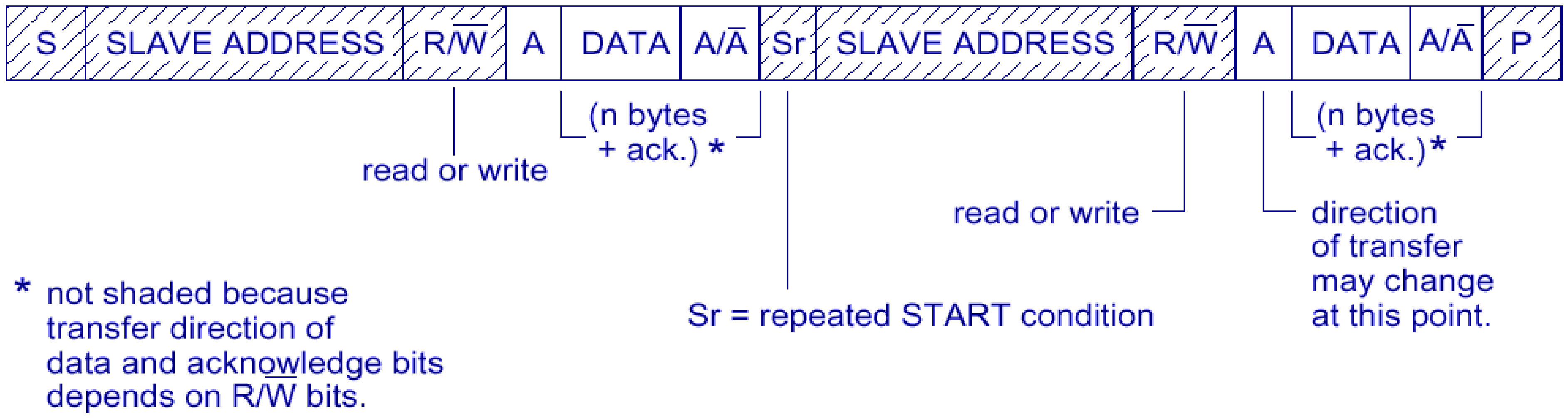
Slave to Master Data Transfer

- Master initiates the data transfer by generating the START condition followed by the start byte (with read/write bit set to 1, i.e. read mode)
- After the first ack from the slave, the direction of data changes and the master becomes receiver and slave transmitter.
- The STOP condition is still generated by the master (master sends not-ACK before generating the STOP)



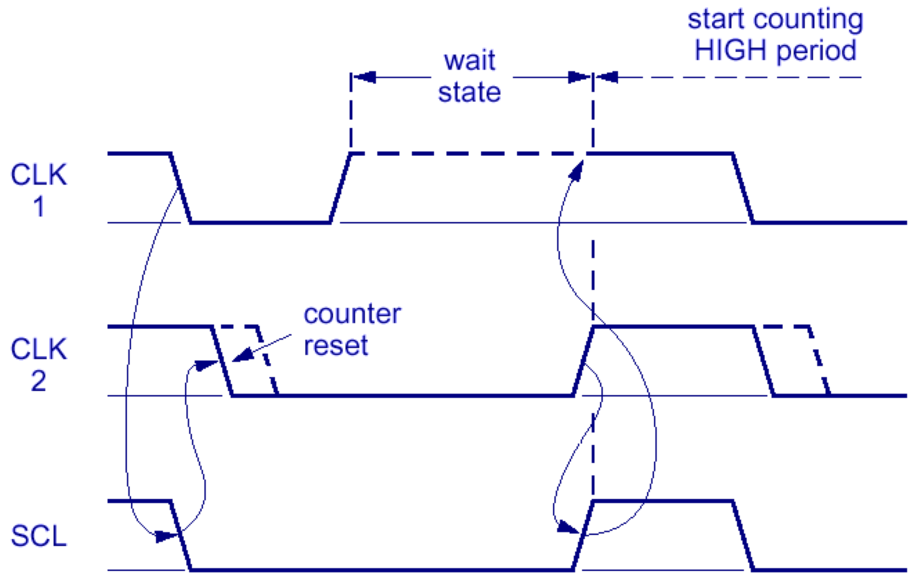
Read and Write in the Same Data Transfer

- Change in direction of data transfer can happen by the master generating another START condition (called the repeated START condition) with the slave address repeated
- If the master was a receiver prior to the change, then the master sends a not-ack (A') before the repeated START condition


S
SLAVE ADDRESS
R/$\overline{W}$
A
DATA
A/$\overline{A}$
Sr
SLAVE ADDRESS
R/$\overline{W}$
A
DATA
A/$\overline{A}$
P
read or write
(n bytes + ack.) *
read or write
(n bytes + ack.)*
direction of transfer may change at this point.
Sr = repeated START condition
* not shaded because transfer direction of data and acknowledge bits depends on R/$\overline{W}$ bits.

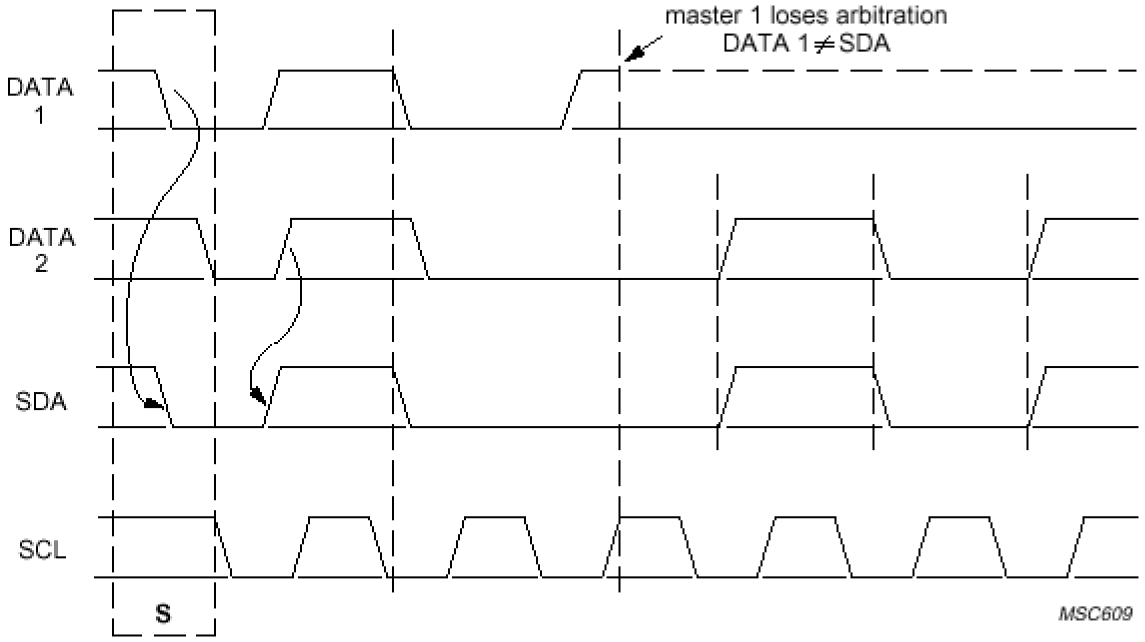
Clock Synchronization

- In the I²C bus, clock synchronization is performed using the wired-AND
  - If at least one master clock goes from HIGH to LOW, then the SCL is held LOW irrespective of the other masters' clock.
  - The SCL line goes to a HIGH state only when all the master clocks are in HIGH
  - Slave can do this as well – "clock stretching"
- The synchronized clock is generated with its LOW period determined by the device with the longest clock LOW period and its HIGH period determined by the one with the shortest clock HIGH period



Multi-Master Arbitration

- If more than one device is capable of being a master, then an arbitration mechanism is needed to choose the master that takes control of the bus
- Arbitration takes place on the SDA, while the SCL is at the HIGH line
  - The master which transmits a HIGH level,
  - Another master is transmitting LOW level will switch off its DATA output stage because the level on the bus does not correspond to its own level

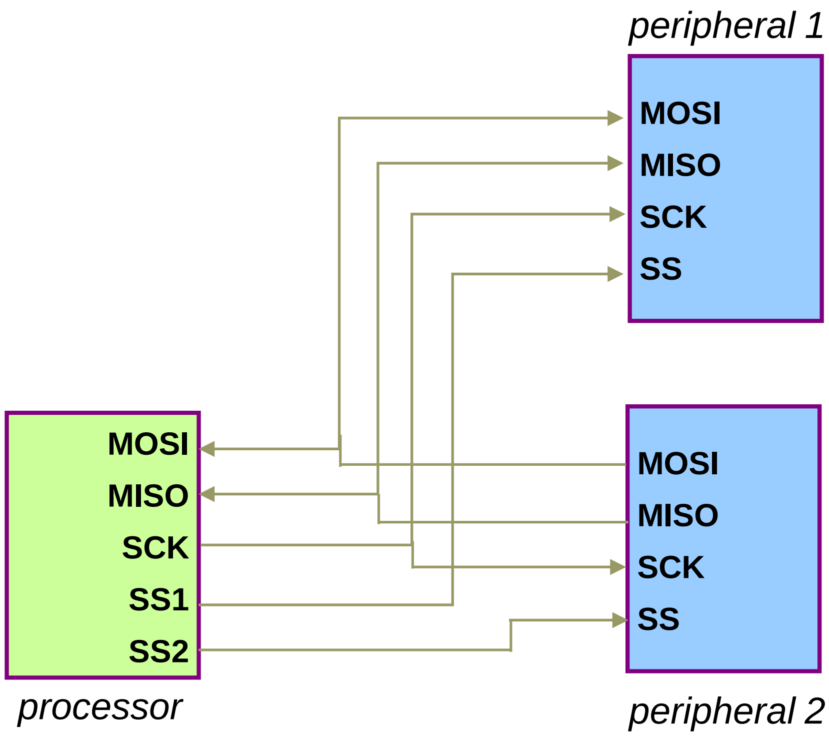


I^2C Summary

- Compared to other serial bus protocols (e.g. SPI, Microwire)
  - The pin (and connection) requirements are the least complex in I^2C
  - The noise immunity is higher for I^2C
  - There is a feedback to the transmitter (Ack signal) for conveying the success of the transfer
  - I^2C now has higher speed modes of operation (~5 Mbit/s)

Serial Peripheral Interface (SPI)

- Defined by Motorola on the MC68HCxx line of microcontrollers
  - Generally faster than I²C, capable of several Mbps
  - Better suited for "data streams", i.e. ADC converters
- A duplex, synchronous, serial communication between CPU and peripheral devices
  - Master mode and slave mode
  - Bi-directional mode
  - Synchronous serial clock
- Signals:
  - MOSI: master out slave in
  - MISO: master in slave out
  - SS: select signal from master to slave
  - SCK: serial clock

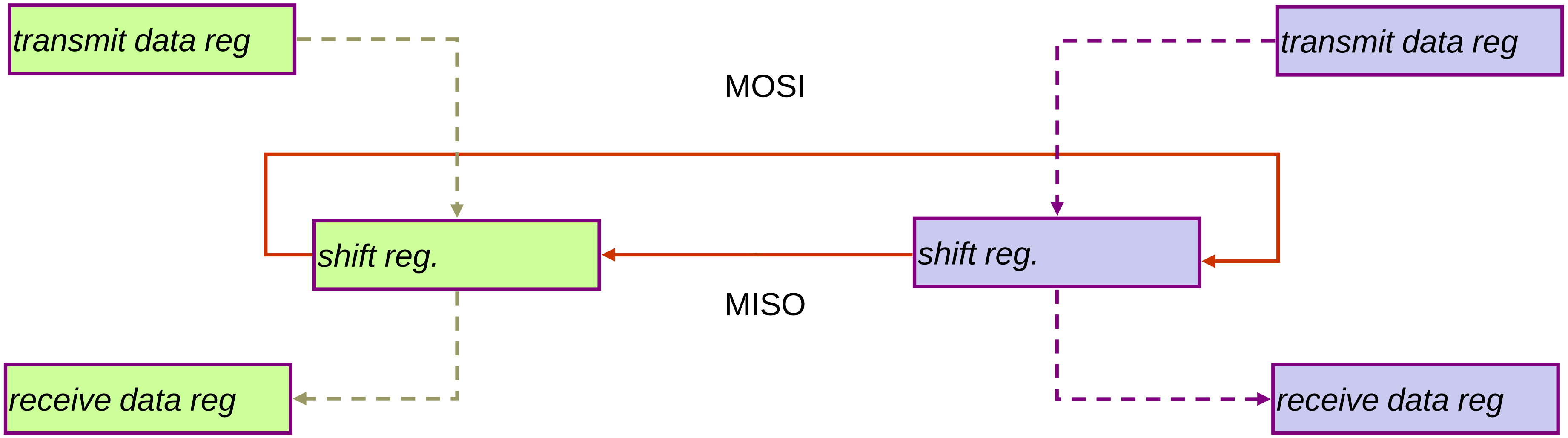


SPI Signaling

- The SPI bus can operate with a single master device and with one or more slave devices
- If a single slave device is used, the SS pin may be fixed to logic low
- Most slave devices have tri-state outputs so their MISO signal becomes high impedance (logically disconnected) when the device is not selected
  - Devices without tri-state outputs can't share SPI bus segments with other devices
  - Only one such slave could talk to the master, and only its chip select could be activated

SPI Operation

- Data registers in the master and the slave form a distributed register
- When a data transfer operation is performed, this distributed register is serially shifted by the SCK clock from the master
- Can shift in burst mode

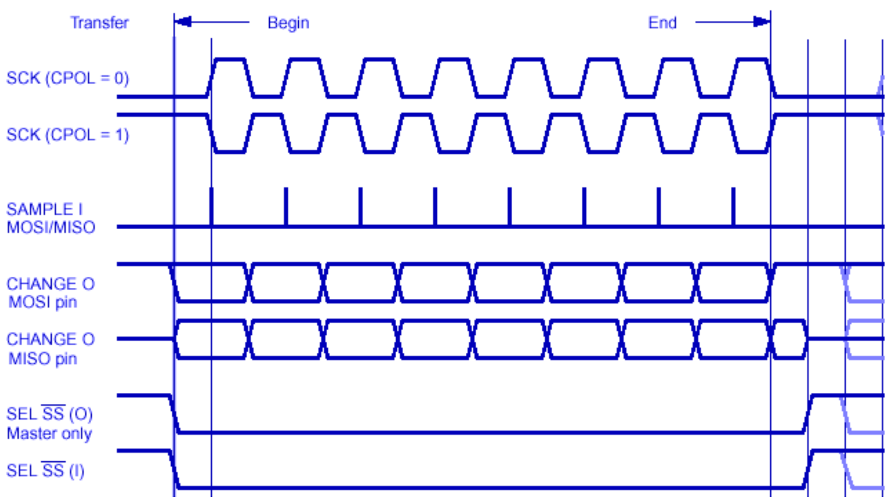


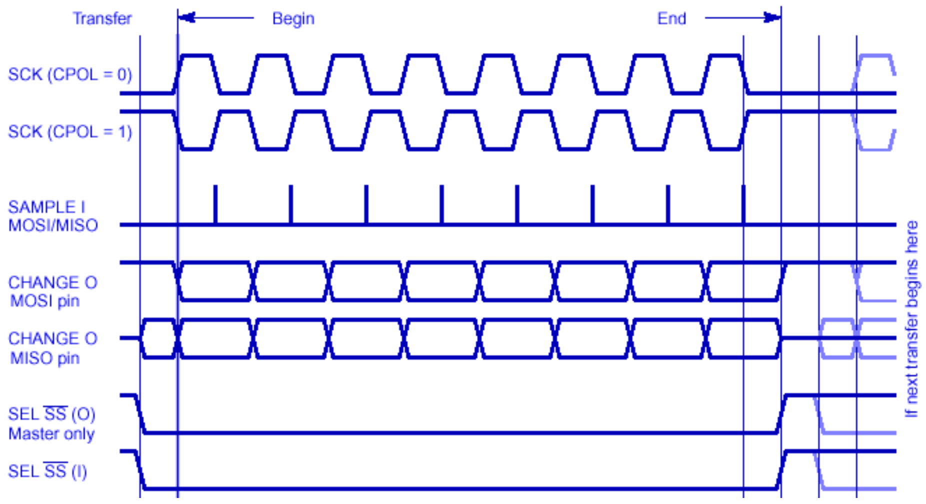
Data Transmission

- To begin a communication, the bus master first configures the clock, using a frequency less than or equal to the maximum frequency the slave device supports. Such frequencies are commonly in the range of 10 kHz–100 MHz
- The master then transmits the logic 0 for the desired chip over the chip select line. A logic 0 is transmitted because the chip select line is active low, meaning its off state is a logic 1; on is asserted with a logic 0
- If a waiting period is required (such as for analog-to-digital conversion), then the master must wait for at least that period of time before starting to issue clock cycles

CPOL and CPHA (Polarity and Phase)

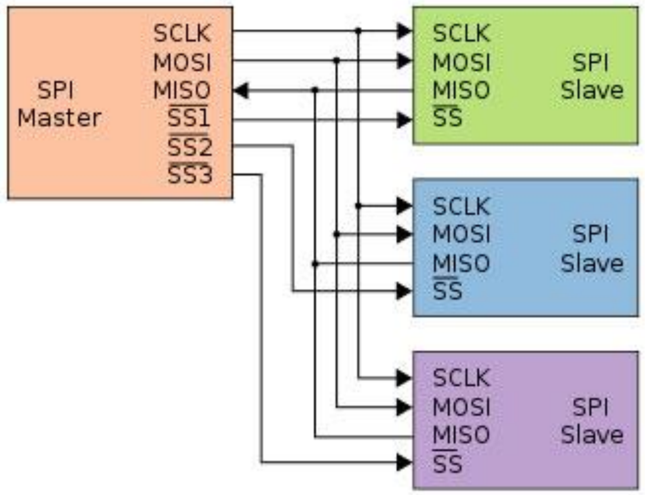
- Two phases and two polarities (four clocking modes)
- CPHA=0 – the first edge on the SCK line is used to clock the first data bit (the first bit of the data must be ready when selected)
- CPHA=1 – if required, the first SCK edge before the first data bit becomes available at the data out pin



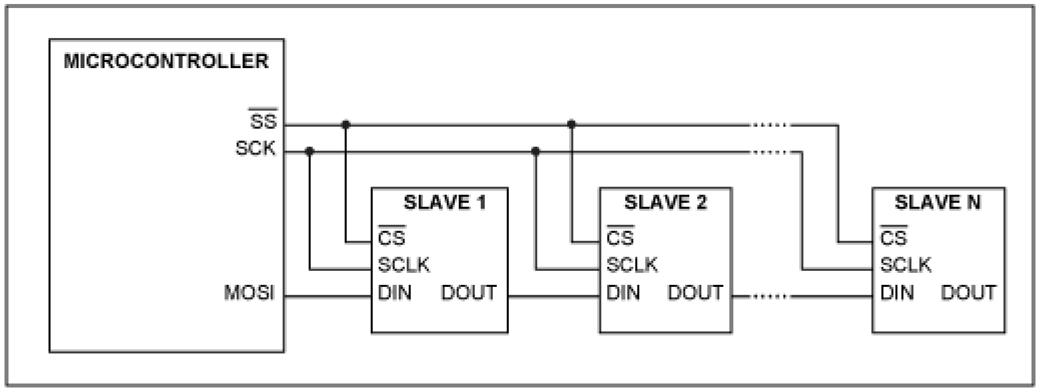


Bus Configuration Modes

- Master and multiple independent slaves



- Master and daisy-chained slaves



SPI Summary

- Pros
  - Fast for point-to-point connections
  - Easily allows streaming/constant data inflow
  - No addressing in protocol, so it's simple to implement
  - Broadly supported
- Cons
  - Slave select/chip select makes multiple slaves more complex
  - No acknowledgement (can't tell if clocking in garbage)
  - No inherent arbitration
  - No flow control (must know slave speed)

Vehicles as Networks

- 1/3 of cost of car/airplane is electronics/avionics
- Dozens of microprocessors are used throughout the vehicle
- Network applications:
  - Vehicle control
  - Instrumentation
  - Communication
  - Passenger entertainment systems

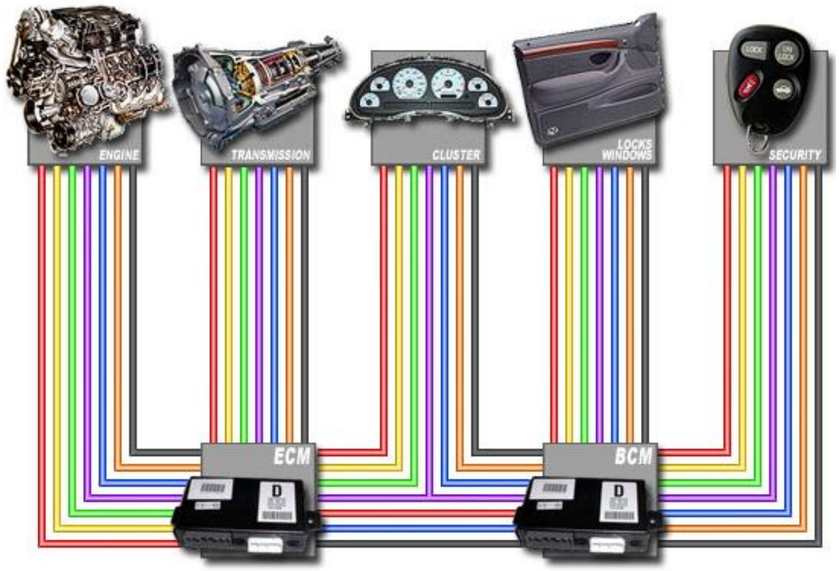
Controller Area Network (CAN)

- Controller Area Network (CAN) is a fast serial bus that is designed to provide
  - an efficient,
  - reliable and
  - very economical link between sensors and actuators
- Introduced by Bosch in 1980s
- CAN uses a twisted pair cable (dual-wire) to communicate at speeds up to 1Mbit/s (max) with up to 40 devices
- It originally developed to simplify the wiring in automobiles
- CAN (and other fieldbus standards) is now used in machine and factory automation products as well

CAN Bus Physical Layer

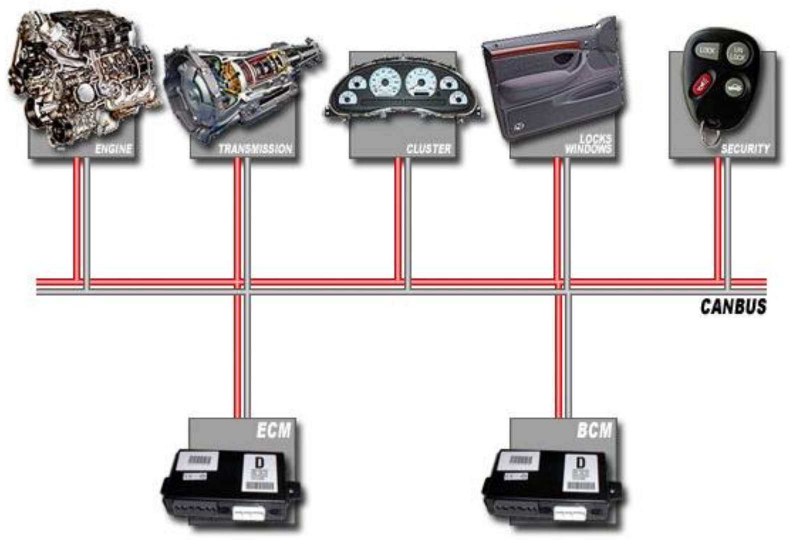
- Physical medium – two wires terminated at both ends by resistors.
  - Differential signal - better noise immunity
  - Benefits:
    - Reduced weight, Reduced cost
    - Fewer wires = Increased reliability

Conventional multi-wire looms

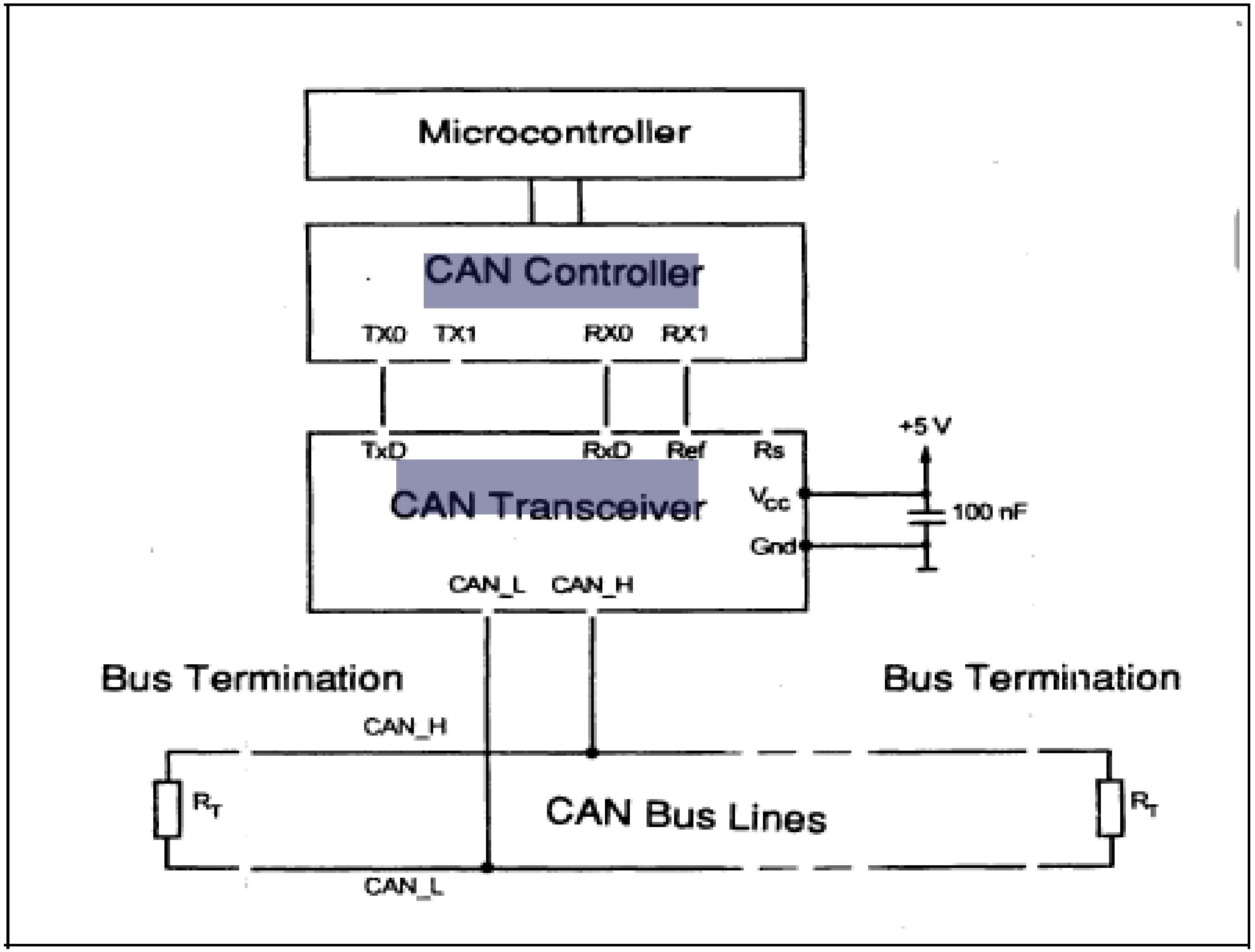


vs.

CAN bus network



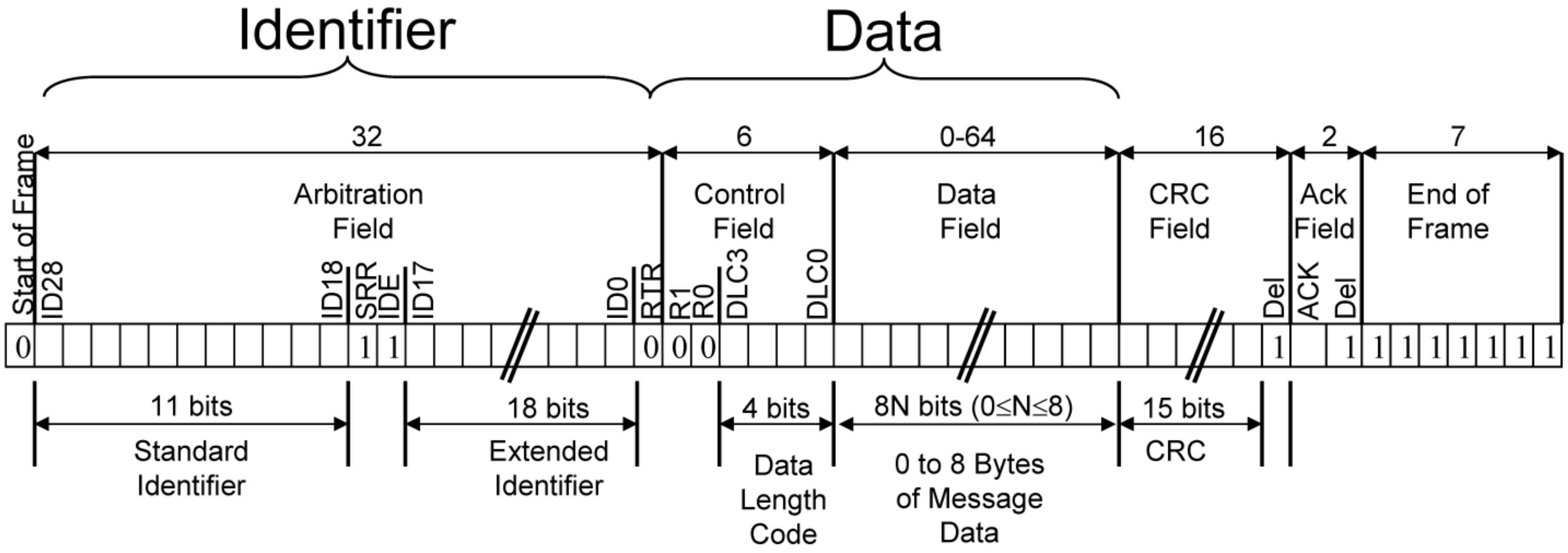
CAN Physical Layer (cont.)



Physical CAN Connection according to ISO 11898

CAN Message Format

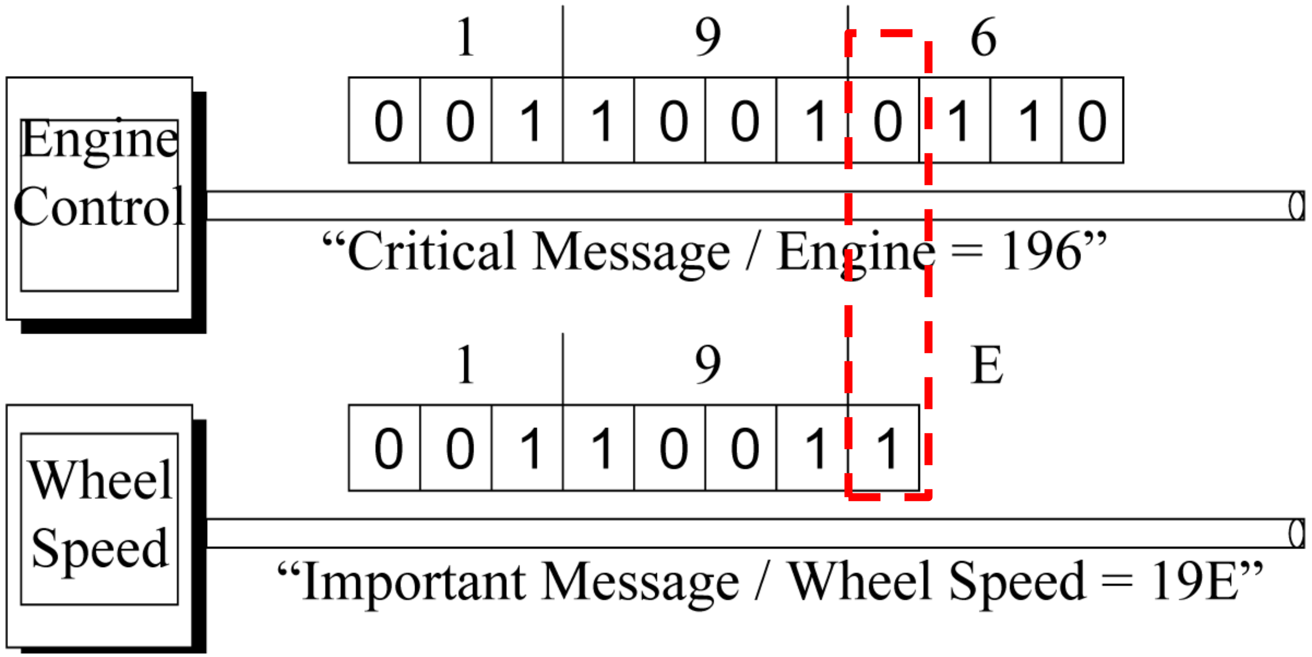
- Each message has an ID, Data and overhead.
- Data –8 bytes max
- Overhead – start, end, CRC, ACK



Bus Arbitration

- Message importance is encoded in message ID

  Lower value = More important
- As a node transmits each bit, it verifies that it sees the same bit value on the bus that it transmitted
- A "0" on the bus wins over a "1" on the bus
- Losing node stops transmitting, winner continues



Engine Control
1
9
6
0 0 1 1 0 0 1 0 1 1 0
0
"Critical Message / Engine = 196"
Wheel Speed
1
9
E
0 0 1 1 0 0 1 1
0
"Important Message / Wheel Speed = 19E"

CAN: Zynq FPGA (from Data Sheet)

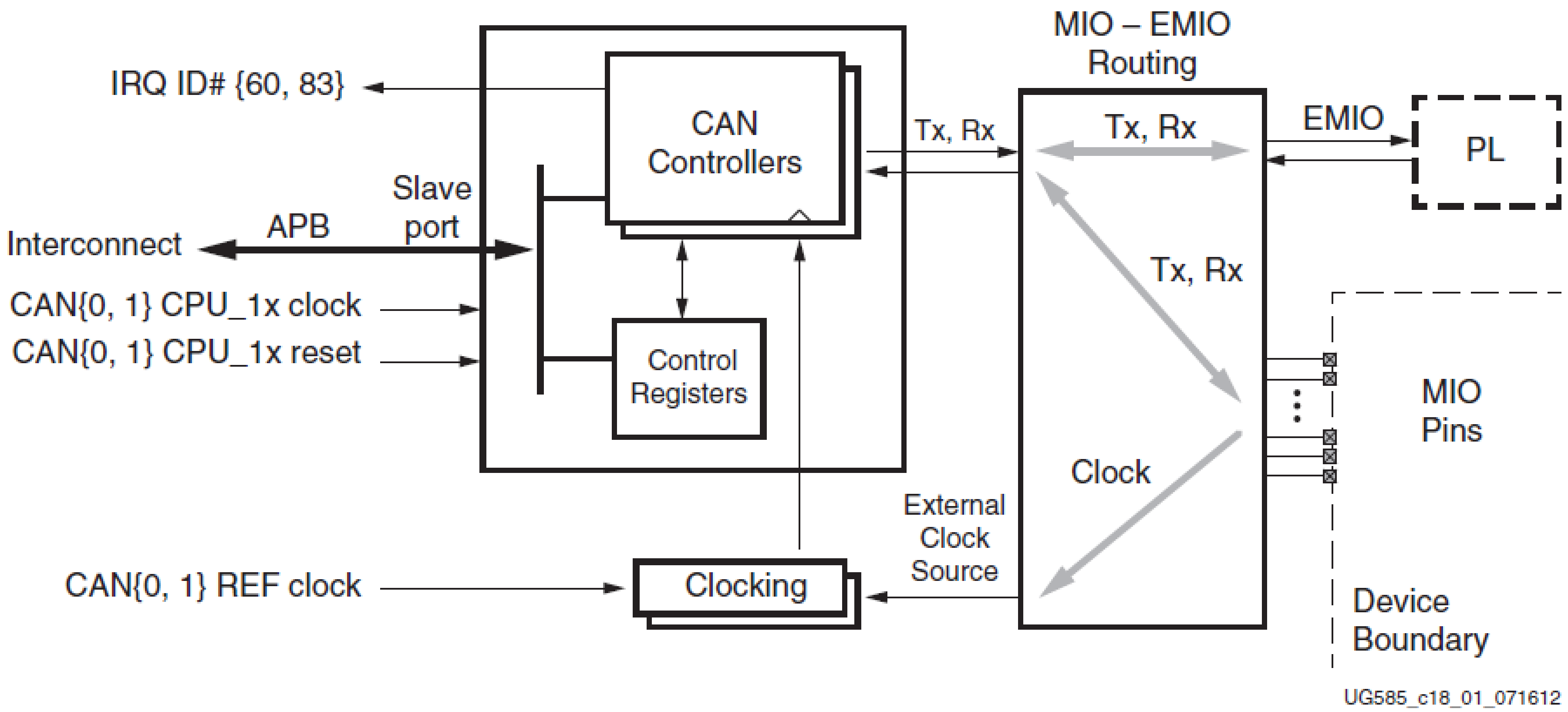


Figure 18-1: **CAN Controller System Viewpoint**

CAN: Zynq FPGA (from Data Sheet)

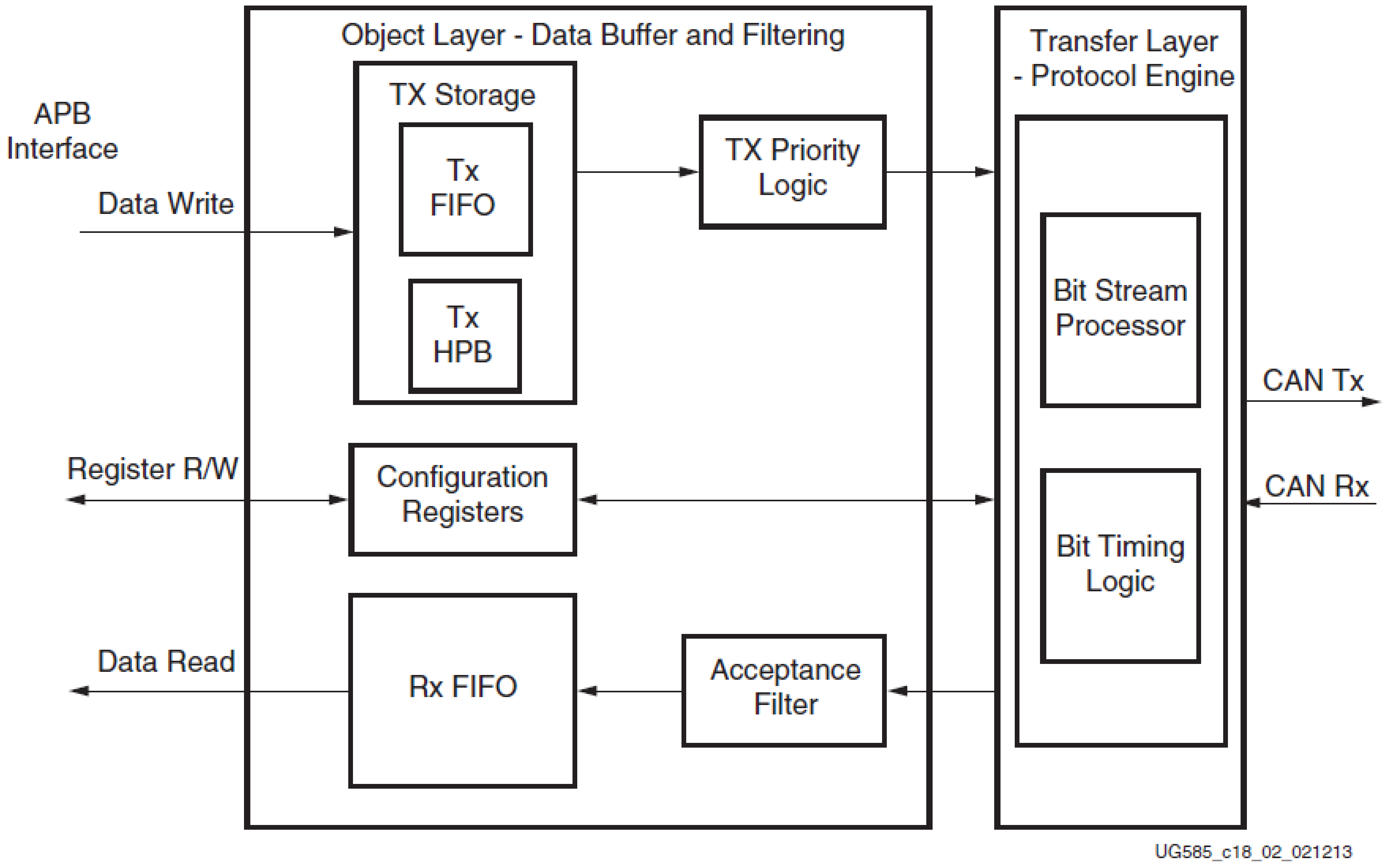


Figure 18-2: **CAN Controller Block Diagram**

CAN: Zynq FPGA (from Data Sheet)

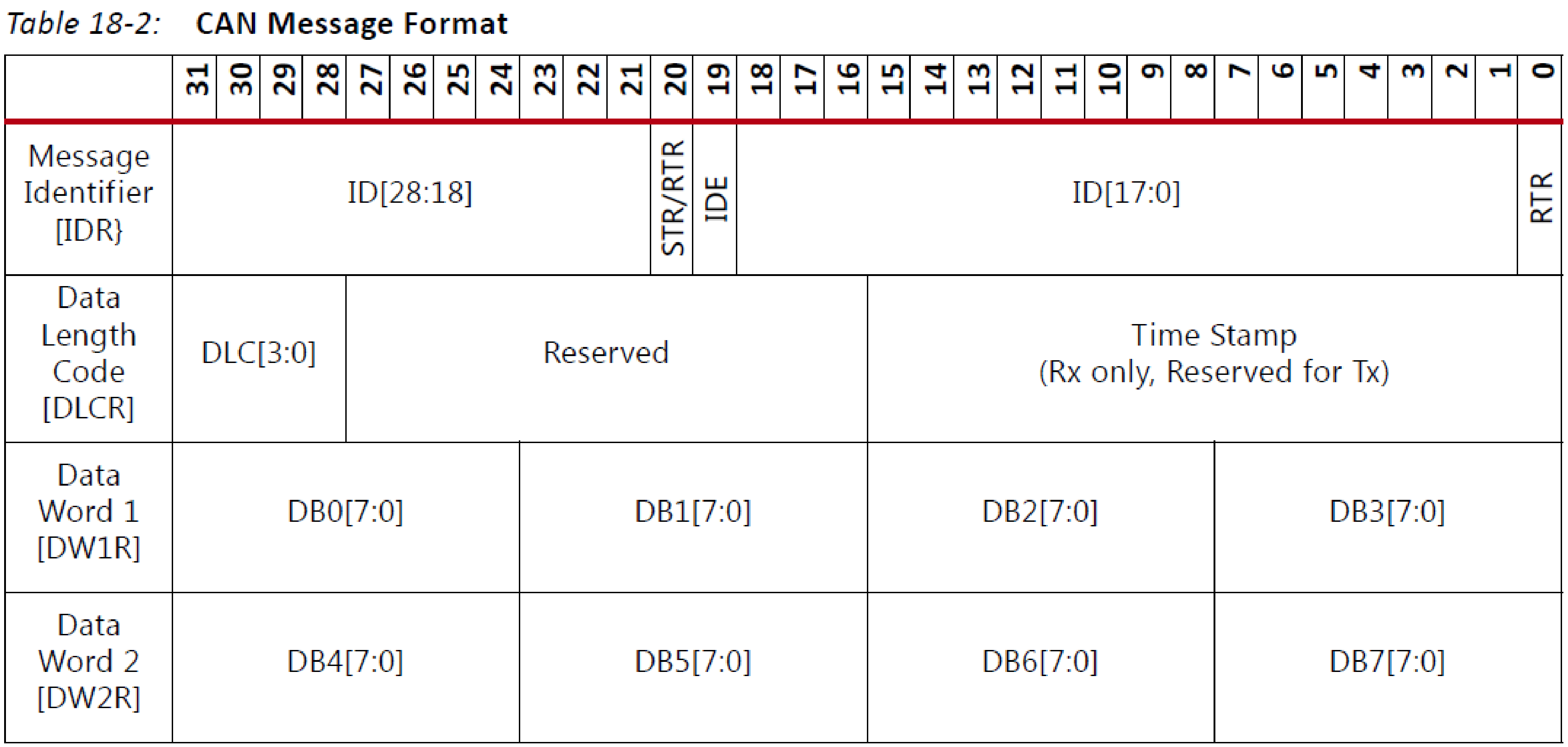


Table 18-2: **CAN Message Format**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Message Identifier [IDR}	ID[28:18]											STR/RTR	IDE	ID[17:0]																		RTR
Data Length Code [DLCR]	DLC[3:0]				Reserved												Time Stamp (Rx only, Reserved for Tx)															
Data Word 1 [DW1R]	DB0[7:0]								DB1[7:0]								DB2[7:0]								DB3[7:0]							
Data Word 2 [DW2R]	DB4[7:0]								DB5[7:0]								DB6[7:0]								DB7[7:0]							

CAN: Zynq FPGA (from Data Sheet)

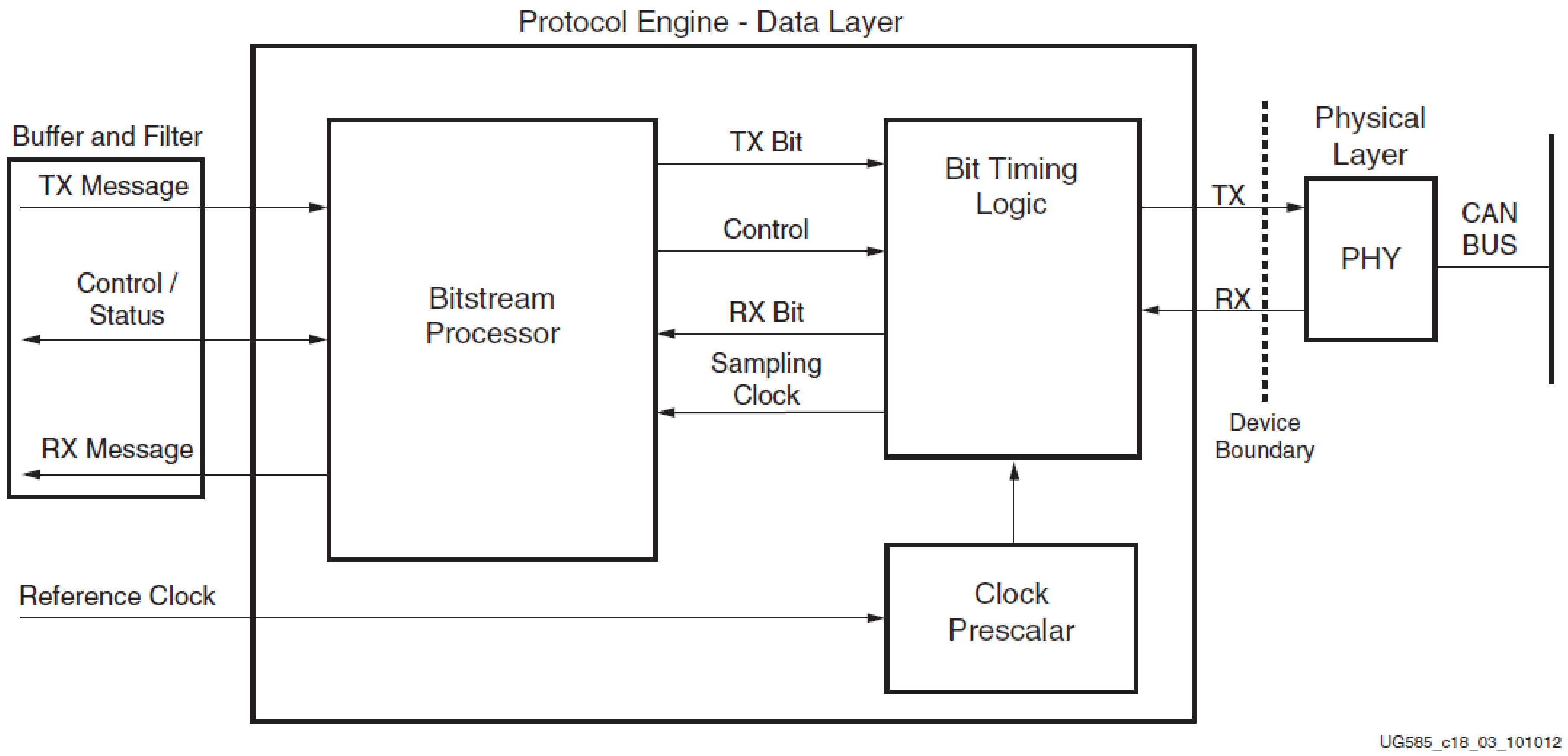


Figure 18-5: **CAN Protocol Engine**

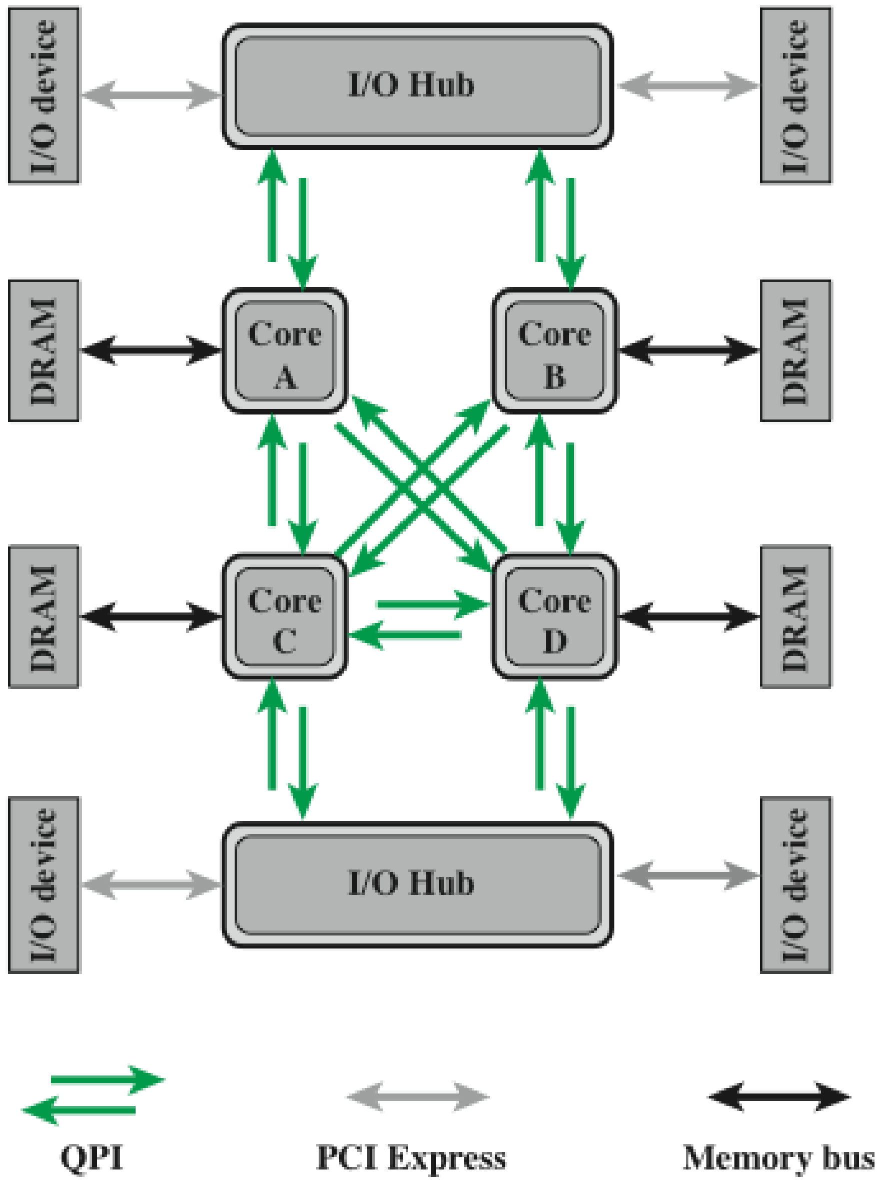
CAN Summary

- CAN bus – **C**ontroller **A**rea **N**etwork bus
- Primarily used for building ECU networks in automotive applications
- Two wires
- OSI - Physical and Data link layers
- Differential signal - noise immunity
- 1Mbit/s, 120’
- Messages contain up to 8 bytes of data

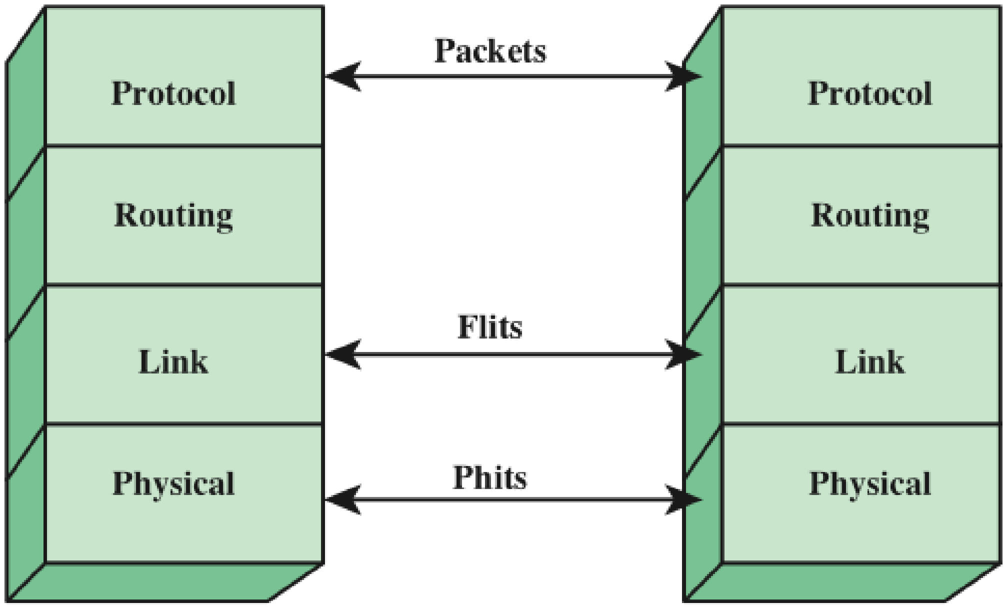
Quick Path Interconnect (QPI)

- Introduced by Intel in 2008 for board-level interfacing
- Multiple direct connections
  - Direct pairwise connections to other components eliminating the need for arbitration found in shared transmission systems
- Layered protocol architecture
  - These processor level interconnects use a layered protocol architecture rather than the simple use of control signals found in shared bus arrangements
- Packetized data transfer
  - Data are sent as a sequence of packets each of which includes control headers and error control codes

Multicore Configuration using QPI

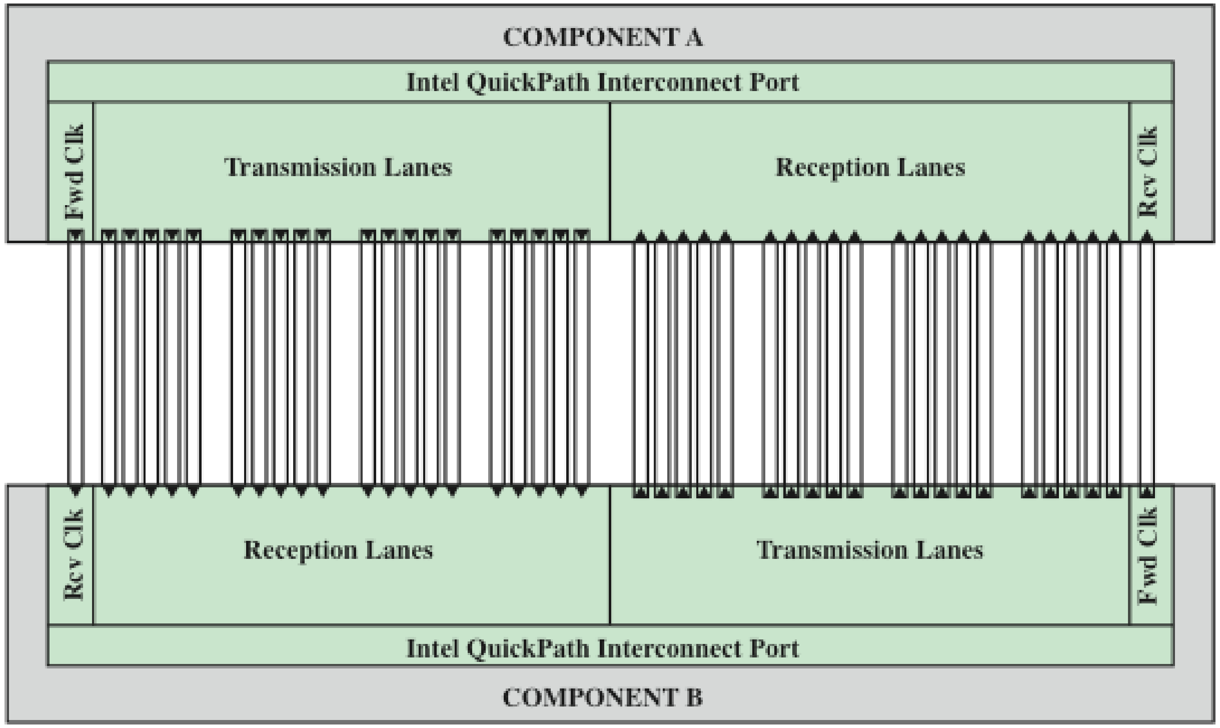


QPI Layers



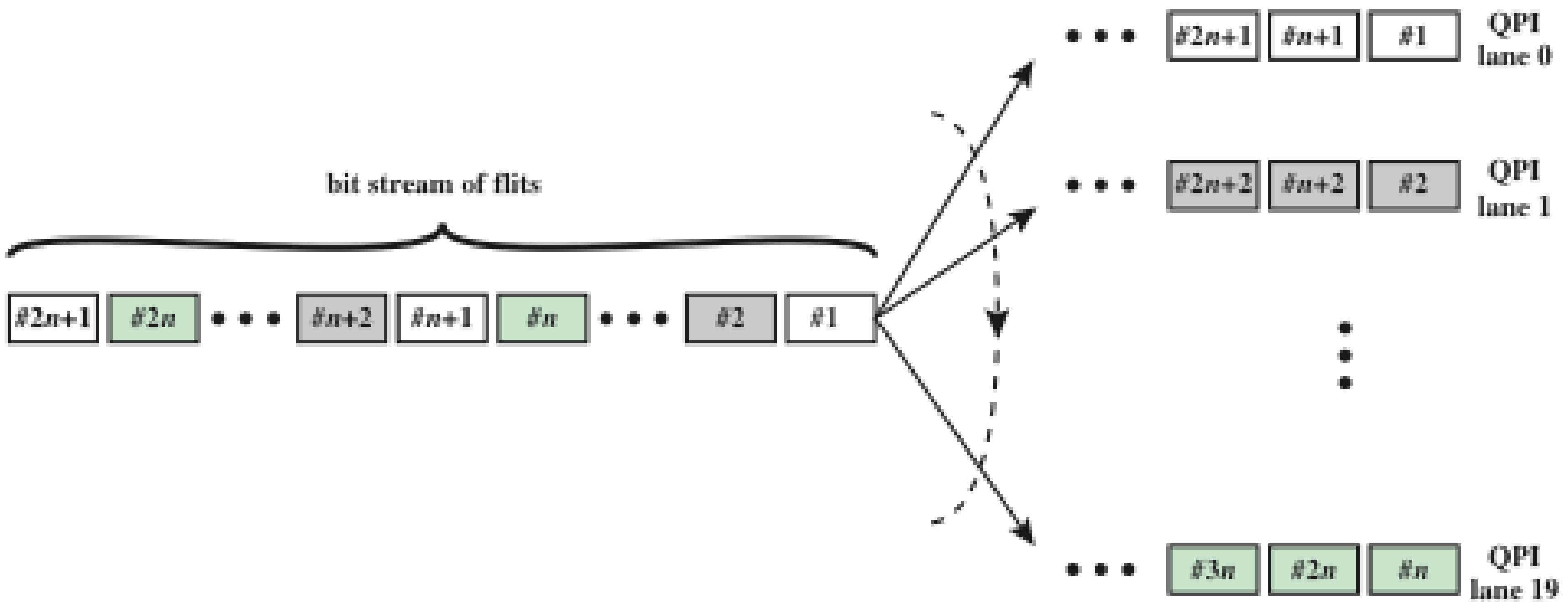
- **Physical:** Consists of the actual wires carrying the signals, as well as circuitry and logic to support ancillary features required in the transmission and receipt of the 1s and 0s. The unit of transfer at the Physical layer is 20 bits, which is called a **Phit** (physical unit).
- **Link:** Responsible for reliable transmission and flow control. The Link layer's unit of transfer is an 80-bit **Flit** (flow control unit).
- **Routing:** Provides the framework for directing packets through the fabric
- **Protocol:** The high-level set of rules for exchanging **packets** of data between devices. A packet is comprised of an integral number of Flits.

QPI Physical Interface



- 20 data lanes in each direction (transmit and receive), plus a clock lane in each direction.
- Typical signaling speeds of the link in current products calls for operation at 6.4 GT/s (transfers per second).
- What is the total bandwidth capacity?

QPI Multilane Distribution



- The flits are distributed across the data lanes in a round-robin fashion
- This approach enables QPI to achieve very high data rates by implementing the physical link between two ports as multiple parallel channels

QPI Link Layer

- Performs two key functions: flow control and error control
  - Operate on the level of the flit (flow control unit)
  - Each flit consists of a 72-bit message payload and an 8-bit error control code called a cyclic redundancy check (CRC)
- Flow control function
  - Needed to ensure that a sending QPI entity does not overwhelm a receiving QPI entity by sending data faster than the receiver can process the data and clear buffers for more incoming data
- Error control function
  - Detects and recovers from bit errors, and so isolates higher layers from experiencing bit errors

QPI Routing and Protocol Layers

- Routing Layer:
  - Used to determine the course that a packet will traverse across the available system interconnects
  - Defined by firmware and describe the possible paths that a packet can follow
- Protocol Layer:
  - Packet is defined as the unit of transfer
  - One key function performed at this level is a cache coherency protocol which deals with making sure that main memory values held in multiple caches are consistent
  - A typical data packet payload is a block of data being sent to or from a cache

Interconnect Summary

- Shared bus architecture is a standard approach for on-chip communication between a processor and other components
- Contemporary systems increasingly rely on point-to-point and serial communication
- Advantages:
  - Simplicity for synchronization and arbitration
  - Electrical constraints
- Many different standards, complexity (e.g. number of wires, corresponding supported OSI-ISO layers) depends on performance requirements

Acknowledgments

- These slides are inspired in part by material developed and copyright by:
  - Marilyn Wolf (Georgia Tech)
  - Yann-Hang Lee (Arizona State)
  - Prabal Dutta (Michigan)
  - Manimaran Govindarasu (Iowa State)
  - William Stallings