# Wanted: Method for Data Sharing among Processors in an Automobile

- Don't know number of processors ahead of time

- Want number of wires between processors to be minimized in order to keep cabling small

- Must be resistant to electrical noise.

- The Controller Area Network (CAN) standard was developed to address this problem
  - ISO-11898 standard define electrical signaling levels to be used for CAN implementation in an automobile.
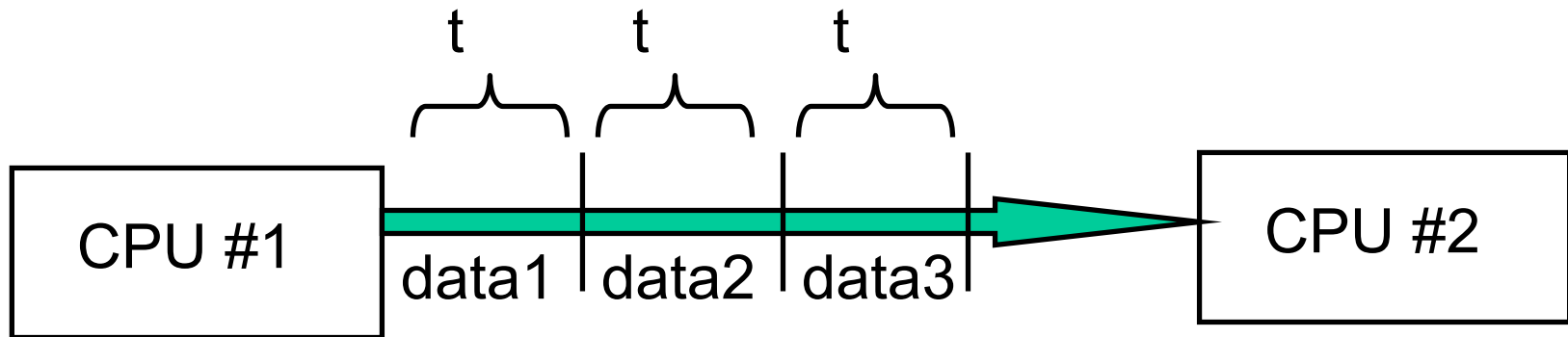
Data Transfer

- Goal: Transfer of data between one or more processors
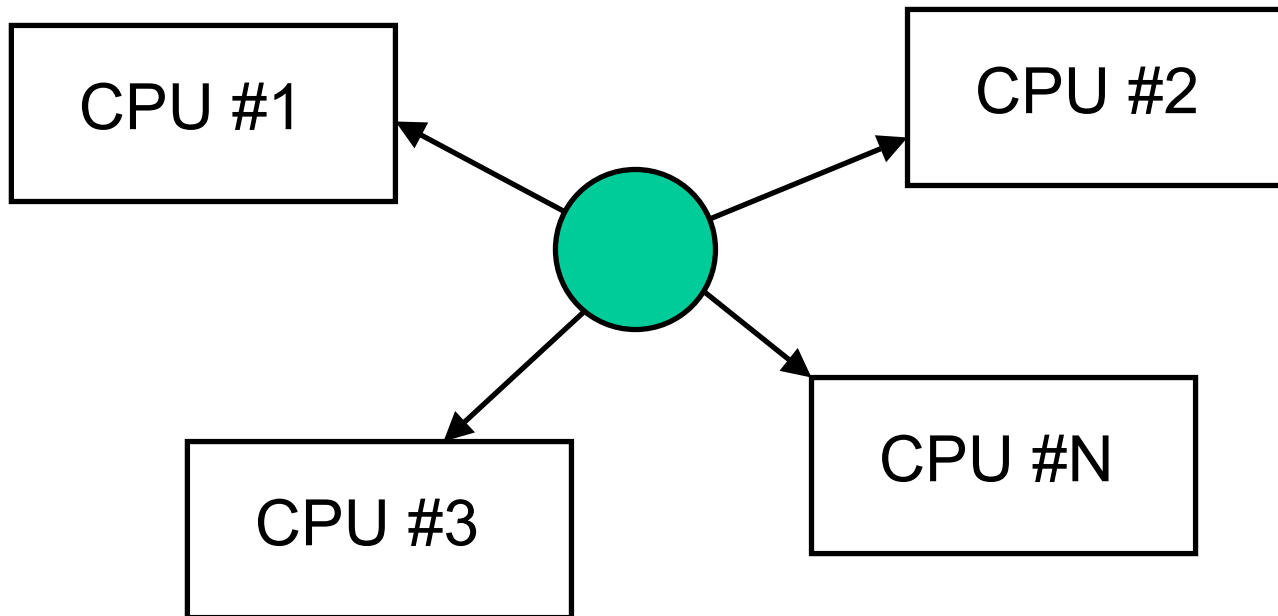
Definition: Bandwidth

t = transfer interval

```
        t        t        t
      ⏞        ⏞        ⏞

┌─────────────┐                              ┌─────────────┐
│             │ ════════════════════►        │             │
│   CPU #1    │ data1 │ data2 │ data3         │   CPU #2    │
│             │                              │             │
└─────────────┘                              └─────────────┘
```

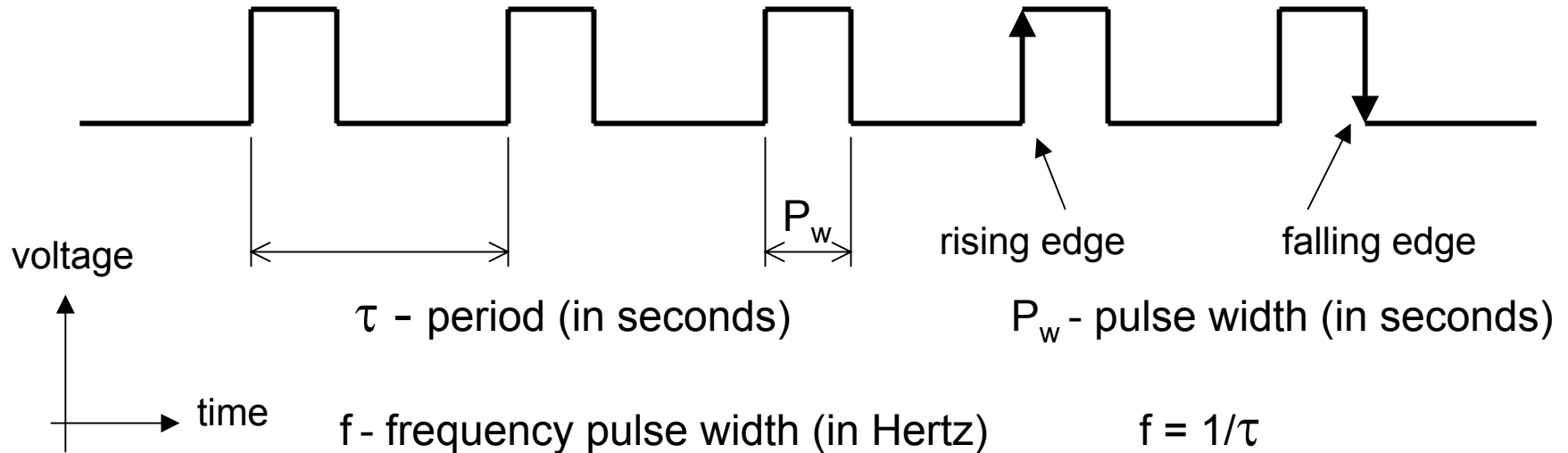Data is typically transferred in fixed time intervals, whose length is agreed upon by both CPUs.

Bandwidth = amount of data/ time interval

I.E.   If four bytes (1 byte = 8 bits) transferred each microsecond (1 us = 1 e-6s), then

bandwidth = 4/ 1e-6 = 4/.000001 = 4,000,000 byte/second

Definition: Clock Signal

Clock: A periodic signal used to measure time.

voltage

time

P_w

rising edge

falling edge

τ - period (in seconds)

P_w - pulse width (in seconds)

f - frequency pulse width (in Hertz)

$f = 1/\tau$

duty cycle - ratio of pulse width to period (in %)

duty cycle $= P_w / \tau$

millisecond (ms) 10^{-3}	Kilohertz (KHz) 10^{3}
microsecond (μs) 10^{-6}	Megahertz (MHz) 10^{6}
nanosecond (ns) 10^{-9}	Gigahertz (GHz) 10^{9}

V 0.3 R. Reese Mar 2006

4

Definition: Parallel Data Transfer

Parallel Transfer – data sent over a group of parallel wires. Typically, a clock is used for synchronization.
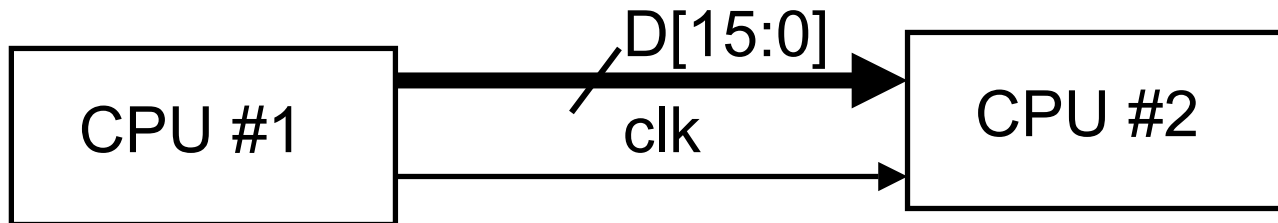

A 16-bit data channel is shown above. If data is transferred each rising clock edge, and clock rate is 300 MHz, then the **data transfer rate (bandwidth)** in bytes/sec is:

2 Bytes/clock period = 2 /(1/300e06)s

$\qquad\qquad\qquad$ = 2 * 300e06/s = 600e06/s

$\qquad\qquad\qquad$ = 600 MB/s   (MB = MBytes)

Definition: Serial Data Transfer

Serial IO – data sent one bit at a time, over a single wire.
A clock may or may not be used for synchronization

D

CPU #1

clk

CPU #2
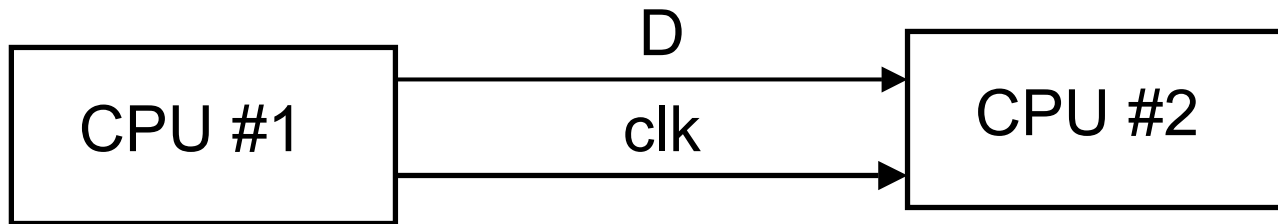
Question: Assuming one bit is sent each rising clock edge, how fast does the clock have to be achieve 600 MB/s?

600 MByte/s = 600 MBytes/s * 8 bits/1Byte = 4800Mb/s

Clock period = 1/4800e06
Clock Frequence = 1/clock period = 4800e06 = 4.8e09 = 4.8GHz

Example of 8-bit Data Transfer (parallel)

D7   D0

Data = 0x8A (hex) = 10001010 (binary)

Parallel Transfer

D7   1

D6   0

D5   0

D4   0

D3   1

D2   0

D1   1

D0   0

CLK

8 data bits transferred on clock edge, takes 8 data lines.
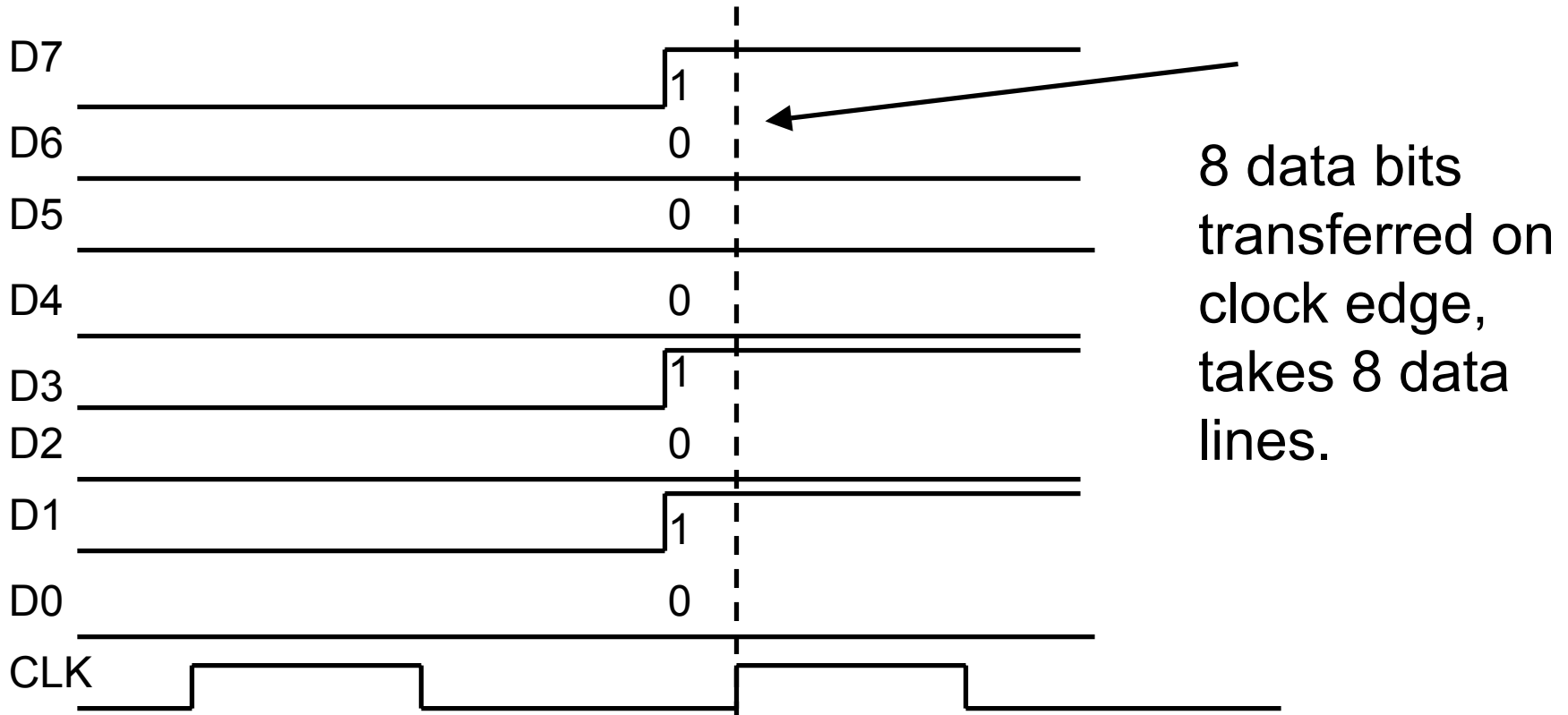
Example of 8-bit Data Transfer (serial)

D7   D0

Data = 0x8A (hex) = 10001010 (binary)

Serial Transfer

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	1	0
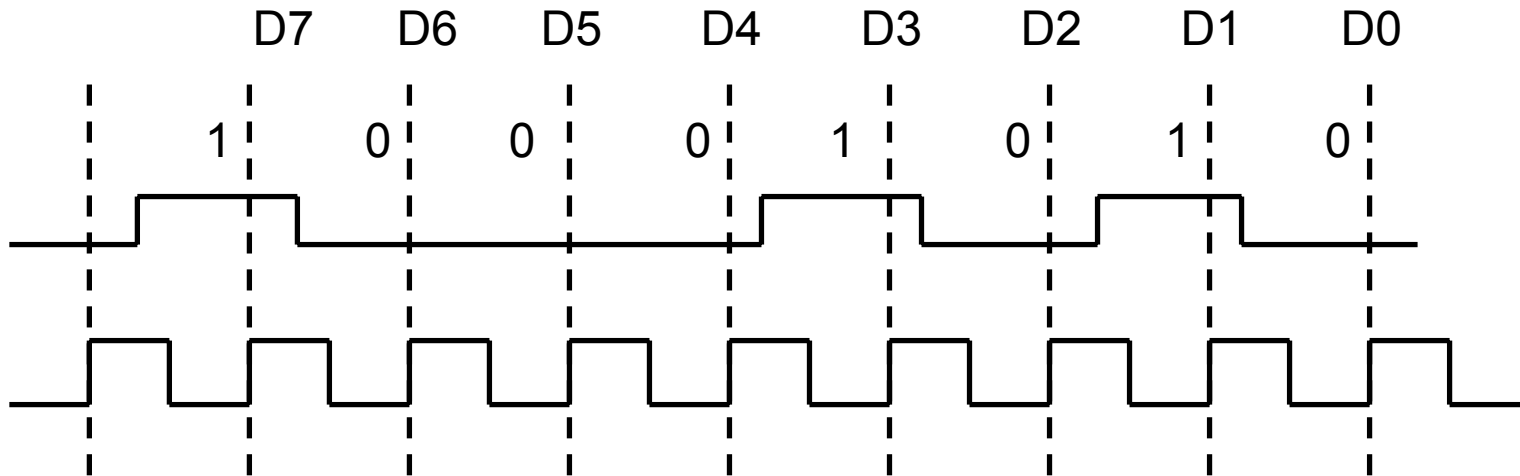
Parallel vs. Serial IO

Used by CAN!

Parallel IO Pros/Cons

Pros: Speed, can increase bandwidth by either making data channel wider or increasing clock frequency

Cons: Expensive (wires cost money!). Short distance only – long parallel wire causes crosstalk, data corruption.

Serial IO Pros/Cons

Pros: Cheap, very few wires needed. Good for long distance interconnect.

Cons: Speed; the fastest serial link will typically have lower bandwidth than the fastest parallel link. However, for long distances (meters), new fast serial IO standards (USB2, Firewire) have replaced older parallel IO standards.

Definitions:simplex vs half-duplex vs full-duplex

For communication channels

CPU #1 → CPU #2
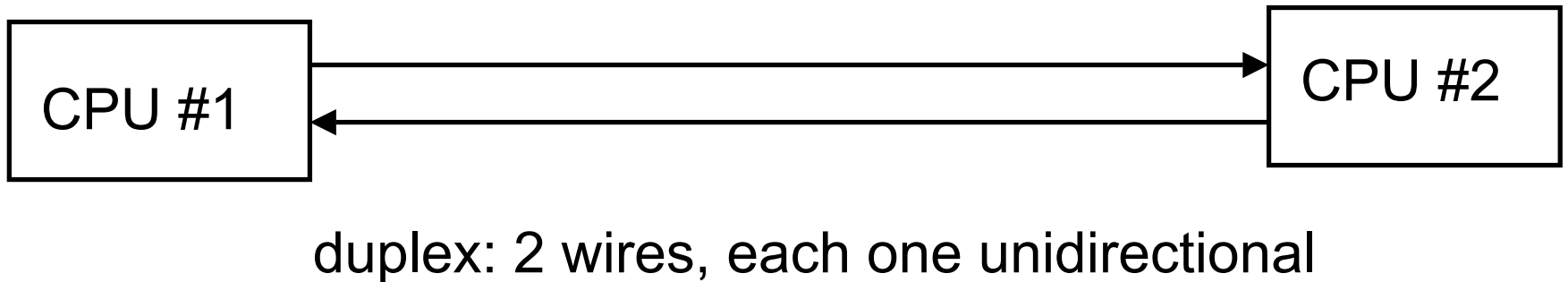
simplex: communication in one direction only

CPU #1 → CPU #2

or

CPU #1 ← CPU #2
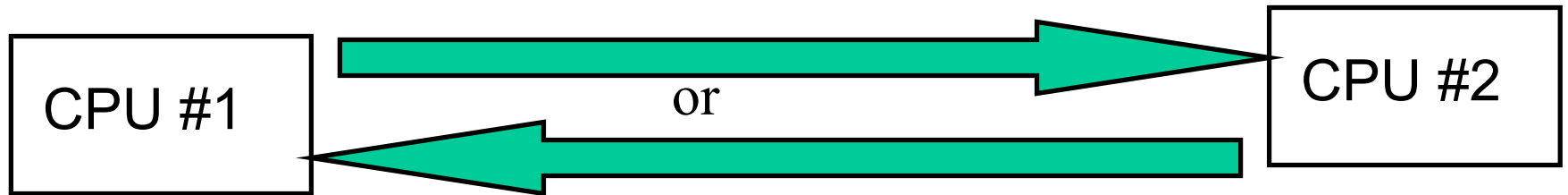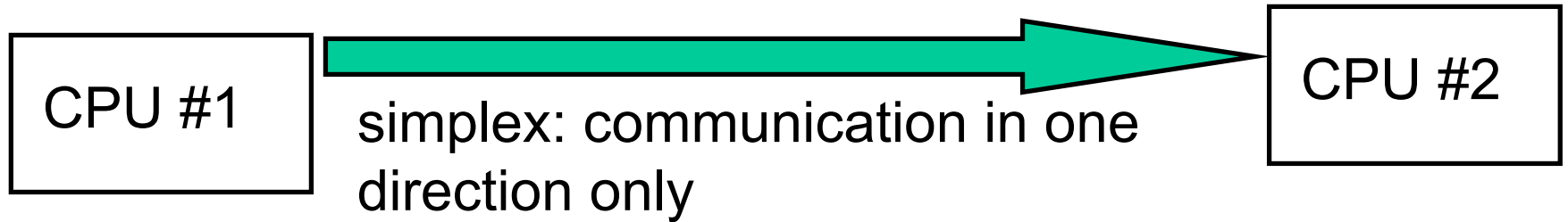
Half-duplex: communication in either direction, but only one way at a time

---

CPU #1 ↔ CPU #2

Full-duplex: communication in both directions at same time.

What is Minimum Number of Wires?

CPU #1 ──────────→ CPU #2

simplex: 1 wire, unidirectional

- -

CPU #1 ←──────────→ CPU #2

half-duplex: 1 wire, that is bidirectional

- -

CPU #1 ──────────→ CPU #2
       ←──────────

duplex: 2 wires, each one unidirectional
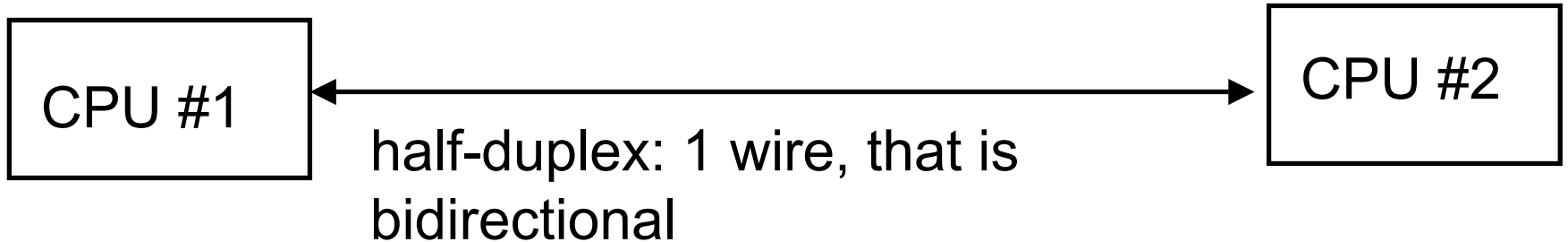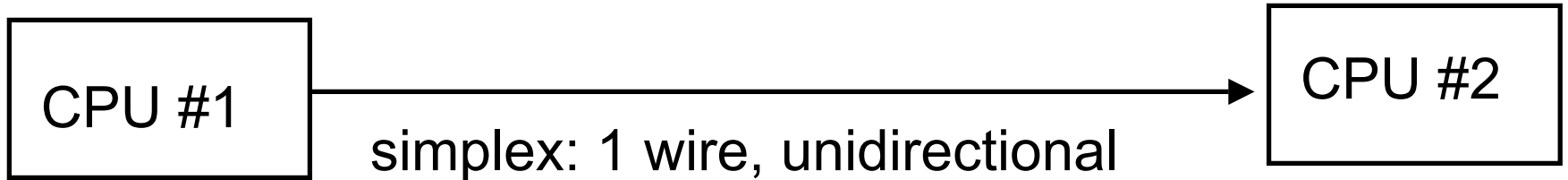
CAN used on automobiles is **HALF-DUPLEX**, and uses **SERIAL DATA TRANSFER**.

Definition: Single-Ended Signaling, Noise Margin

Noise Margin: amount of noise that can be injected into signal without suffering corruption.

CMOS, Vdd= 5 V typical values:

V_{IH} = 2.0 V
V_{IL} = 0.8 V
V_{OH} = 4.3 V
V_{OL} = 0.6 V

driver   one wire used for single-ended signaling   receiver

V_{OH} – logic 1
V_{OL} – logic 0

V_{IH} – logic 1
V_{IL} – logic 0

noise margin , logic 1

V_{OH} – logic 1
V_{IH} – logic 1

$V_{IH} < V_{OH}$

Indeterminate value

noise margin , logic 0

V_{IL} – logic 0
V_{OL} – logic 0

$V_{IL} > V_{OL}$

Voltage

Definition: Double-Ended Signaling

Transmitter   Noise Source Vn   Receiver

Differential signaling is used in CAN.

$$Vo = (Va + Vn) - (Vb + Vn)$$
$$= (Va - Vb) + (Vn - Vn) \rightarrow 0$$
$$= Va - Vb$$

Common-mode noise Vn is rejected
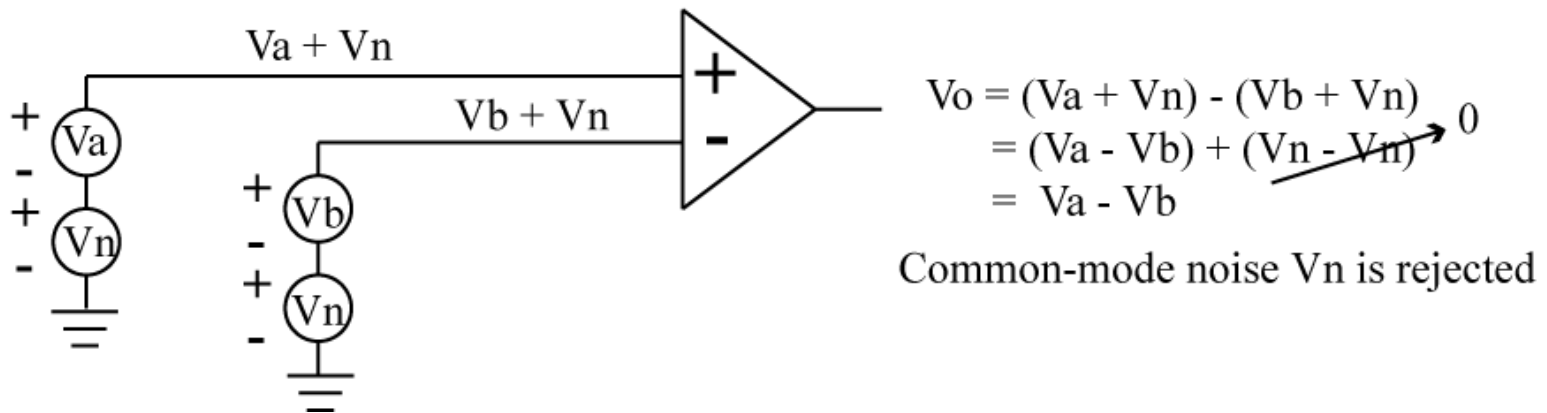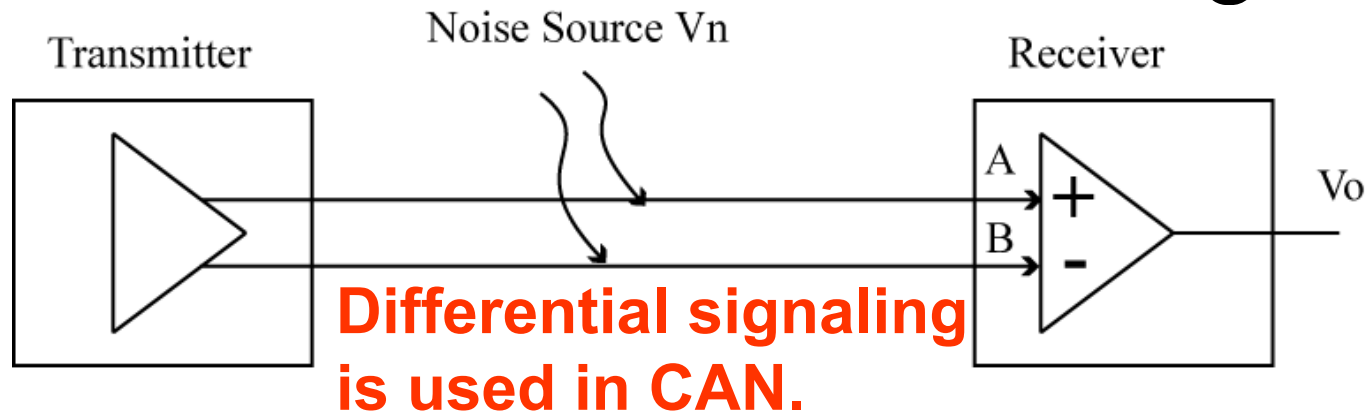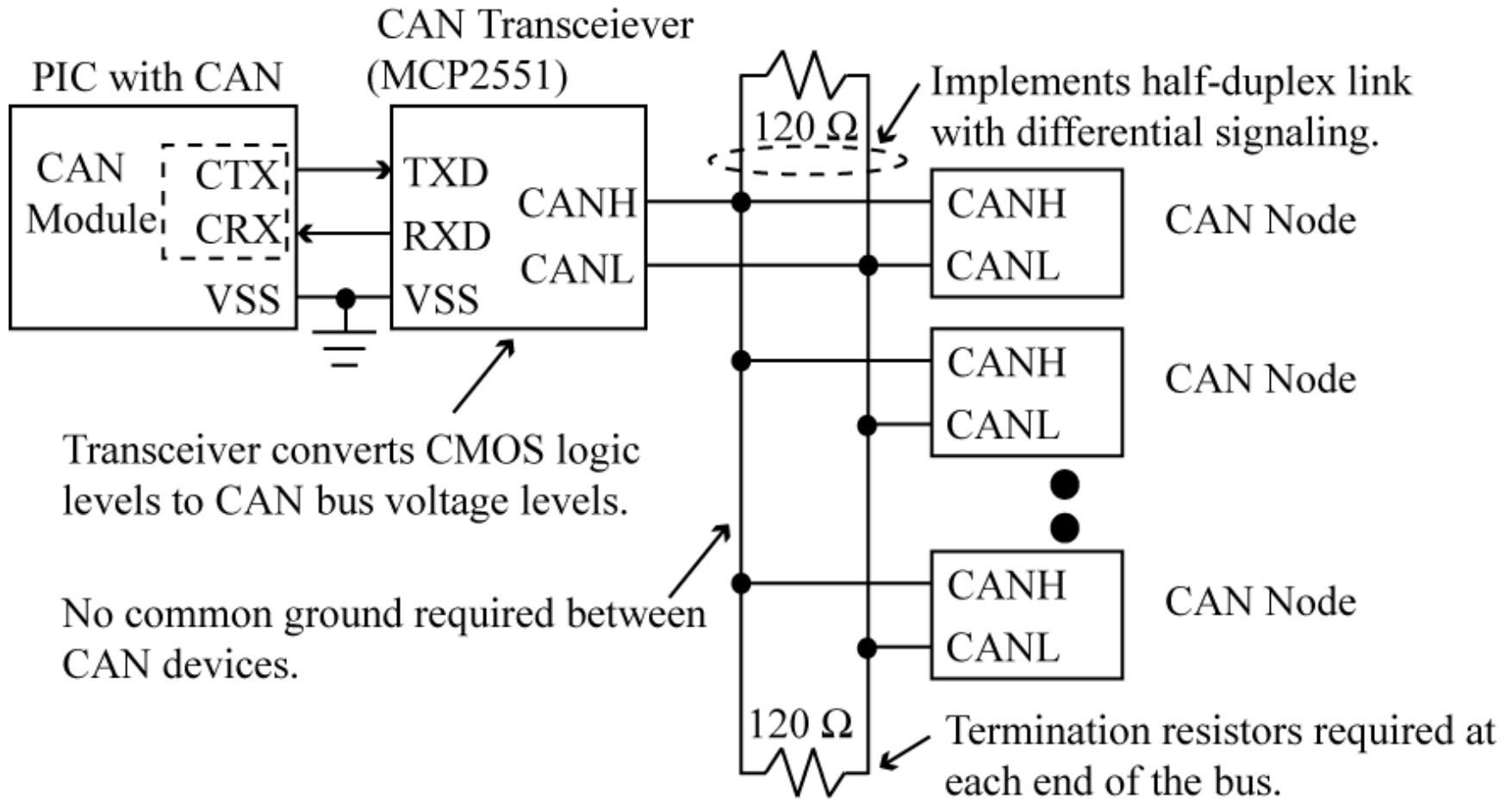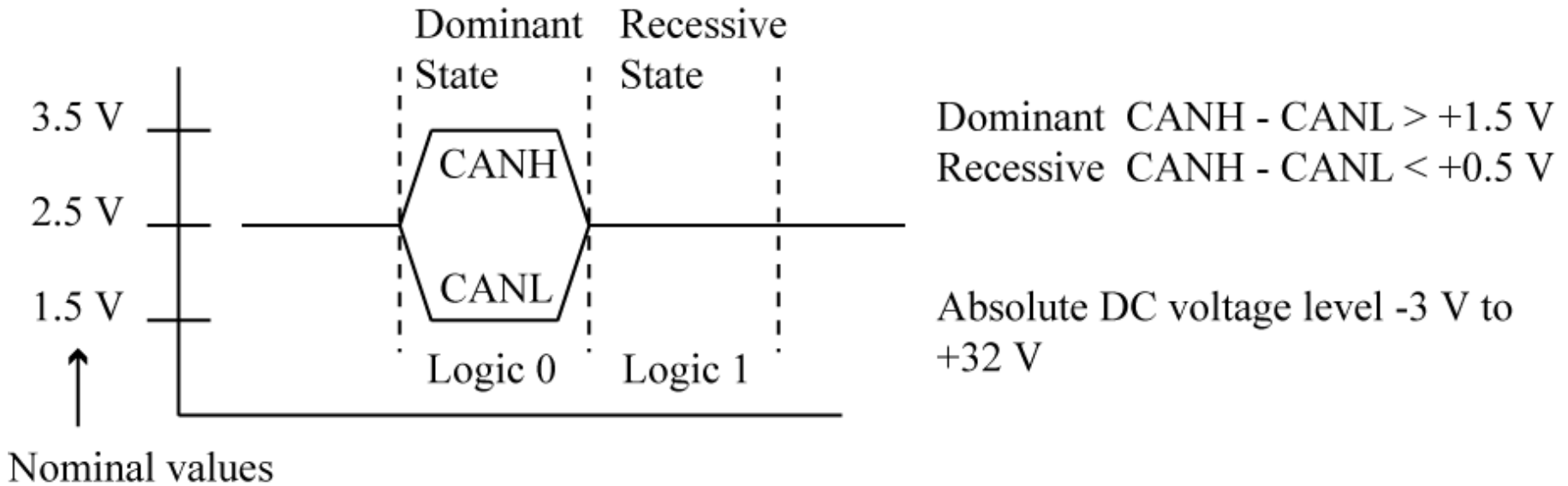
Differential signaling uses two wires to transmit a value. Voltage difference between two wires is the logic '1' or logic '0'. *Common-mode* noise is noise injected into both wires, this is subtracted out at the receiver (is said to be *rejected*).

CAN Interconnect

CAN Transceiever (MCP2551)

PIC with CAN

CAN Module — CTX, CRX, VSS

TXD, RXD, VSS

CANH, CANL

120 Ω — Implements half-duplex link with differential signaling.

CAN Node — CANH, CANL

CAN Node — CANH, CANL

CAN Node — CANH, CANL

120 Ω — Termination resistors required at each end of the bus.

Transceiver converts CMOS logic levels to CAN bus voltage levels.

No common ground required between CAN devices.

CAN Signaling

Dominant State   Recessive State

3.5 V

2.5 V

1.5 V

CANH

CANL

Logic 0   Logic 1

Nominal values

Dominant CANH - CANL > +1.5 V
Recessive CANH - CANL < +0.5 V

Absolute DC voltage level -3 V to +32 V

Only TWO wires are used to transmit information.

A common ground wire is not used, so absolute DC voltage levels between systems can vary widely.

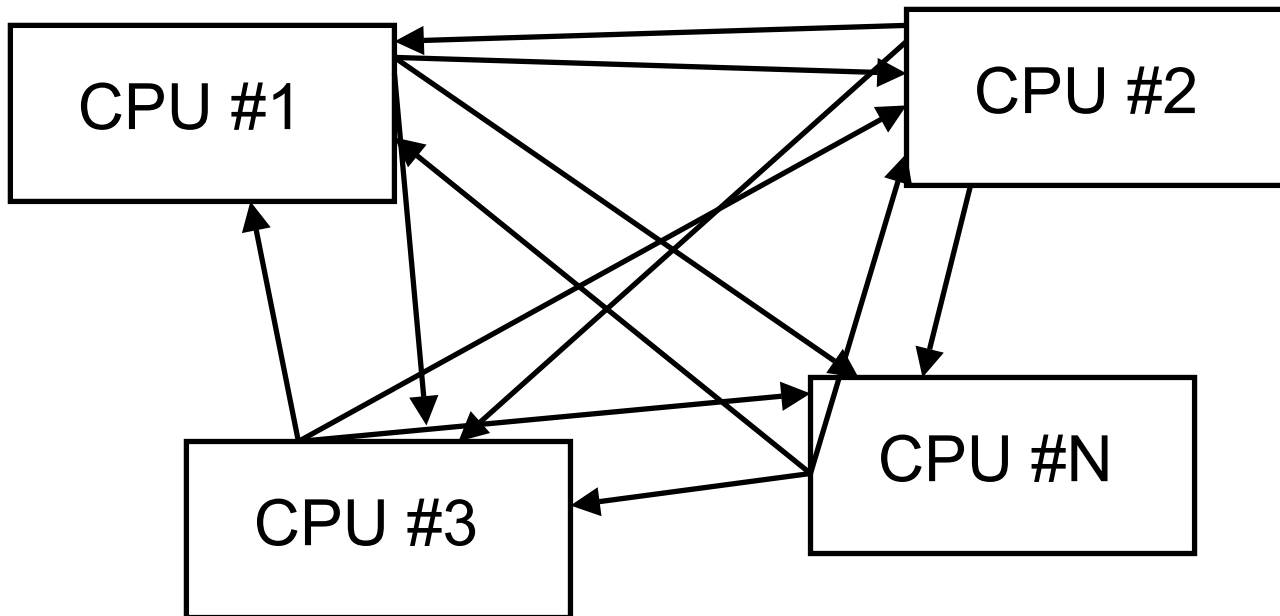Differential signaling makes transmission more noise-resistant.

Connecting multiple processors

CPU #1

CPU #2

CPU #3

CPU #N
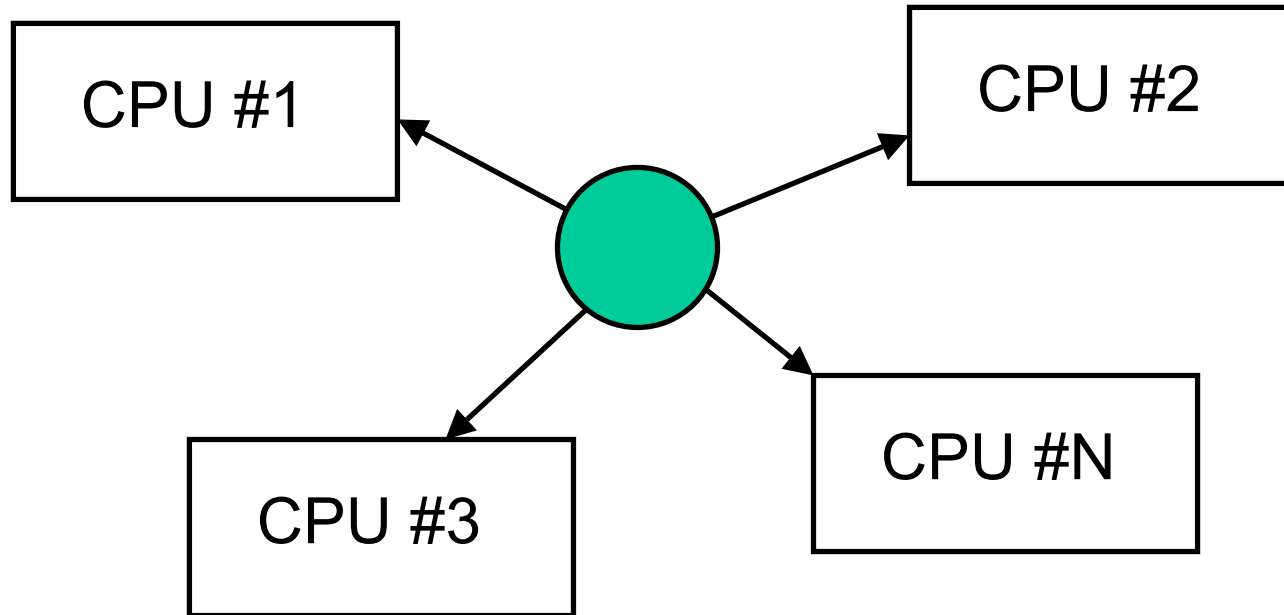
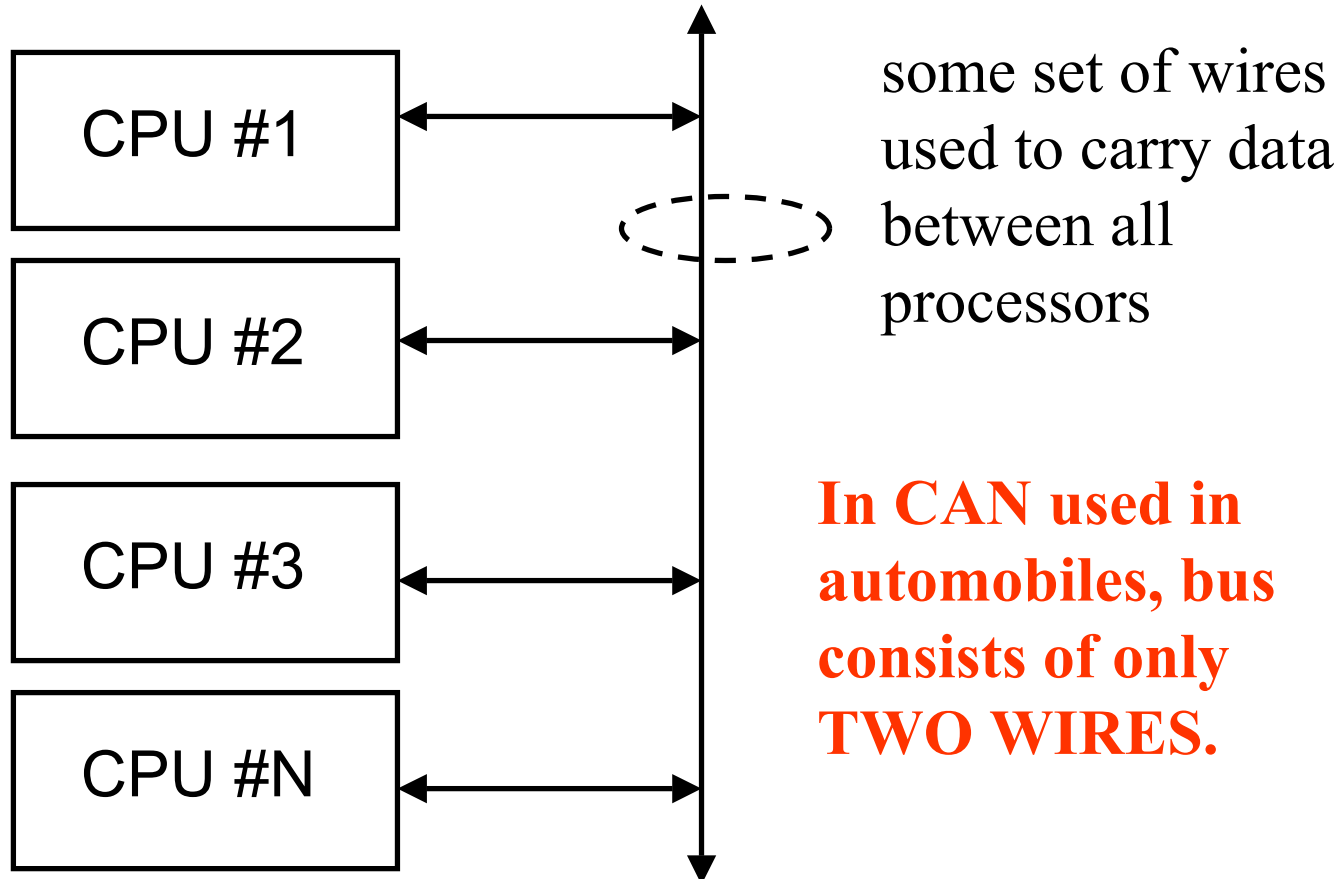How do we connect multiple processors?

Point-to-Point

Dedicated wires between every CPU.

Impractical, too costly.

Definition: Bus

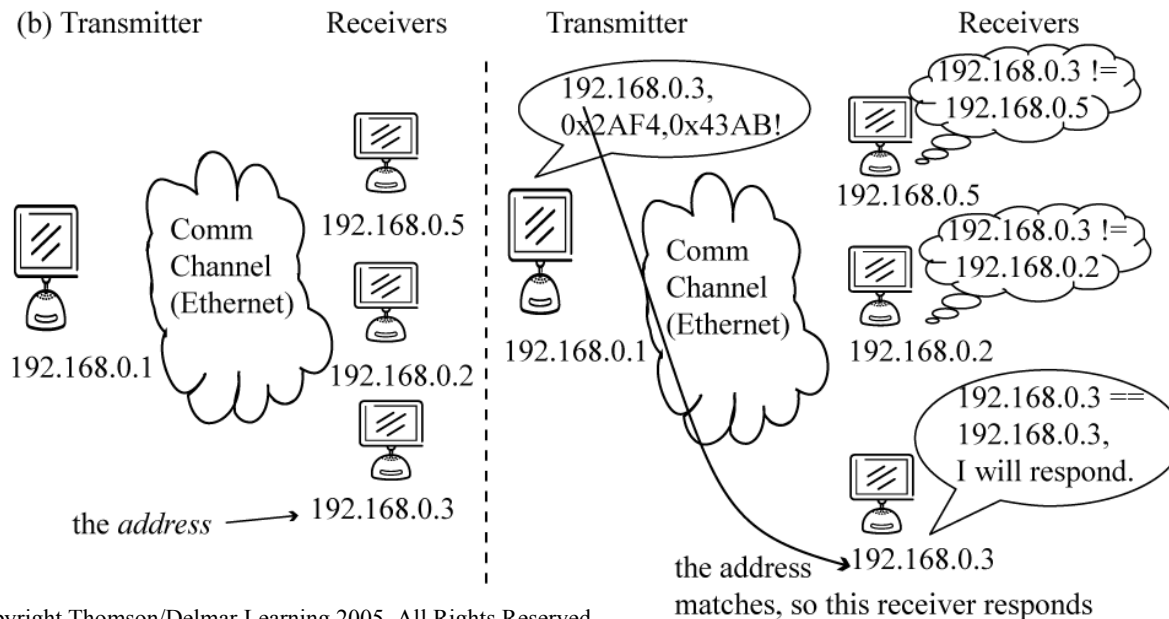*Bus*: A common communication channel between multiple processors.

some set of wires used to carry data between all processors

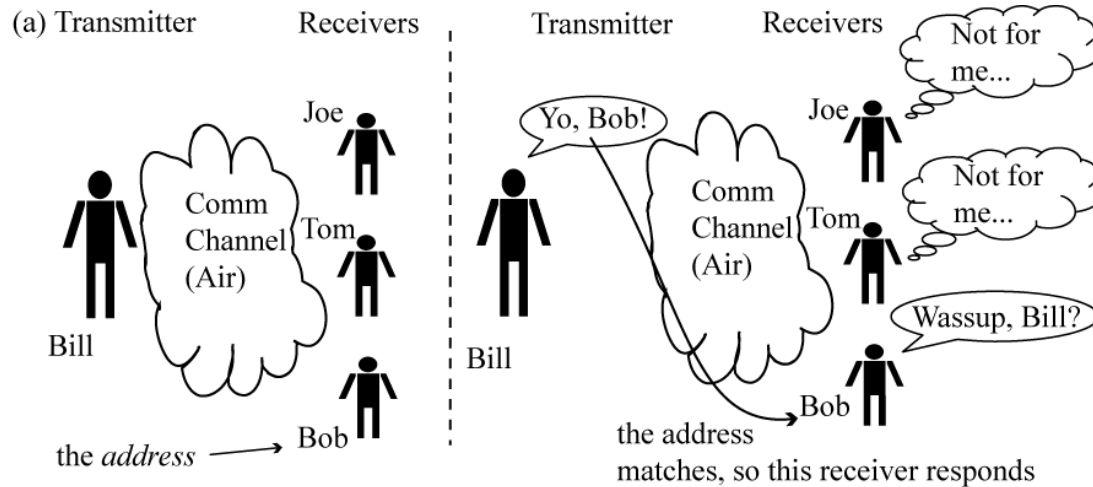**In CAN used in automobiles, bus consists of only TWO WIRES.**

Definition: Transaction

Transaction: Exchange of data between one transmitter and one or more recipients.

Data that is transferred must somehow identify the recipient.

This is called *addressing*.

Bus Addressing

(a) Transmitter / Receivers — Bill transmits to Joe, Tom, Bob over Comm Channel (Air); the *address* → Bob. In the second panel, Bill says "Yo, Bob!"; Joe thinks "Not for me...", Tom thinks "Not for me...", Bob says "Wassup, Bill?" — the address matches, so this receiver responds.

(b) Transmitter / Receivers — 192.168.0.1 transmits over Comm Channel (Ethernet) to 192.168.0.5, 192.168.0.2, 192.168.0.3; the *address* → 192.168.0.3. In the second panel, 192.168.0.1 transmits "192.168.0.3, 0x2AF4,0x43AB!"; 192.168.0.5 thinks "192.168.0.3 != 192.168.0.5", 192.168.0.2 thinks "192.168.0.3 != 192.168.0.2", 192.168.0.3 says "192.168.0.3 == 192.168.0.3, I will respond." — the address matches, so this receiver responds.

One method is to explicitly NAME each receiver with a unique identifier.

Message contains identifier of recipient.

This is NOT USED within CAN.

20

Addressing in CAN

CAN assigns each MESSAGE an identifier.

A processor then monitors the bus, and determines which MESSAGES it wants to receive.

CPU #0 may send messages with identifiers 30, 100, 46.

CPU #1 may elect to receive messages with identifiers of 30 and 46.

CPU #2 may elect to only receive messages with an identifier of 100.

Identifiers in CAN are 11 bits long, giving 2048 unique message identifiers ($2^{11} = 2048$)

Why is addressing in CAN done this way?

Most data on the CAN in an automobile consists of **SENSOR** data that is **periodically** sent by a processor.
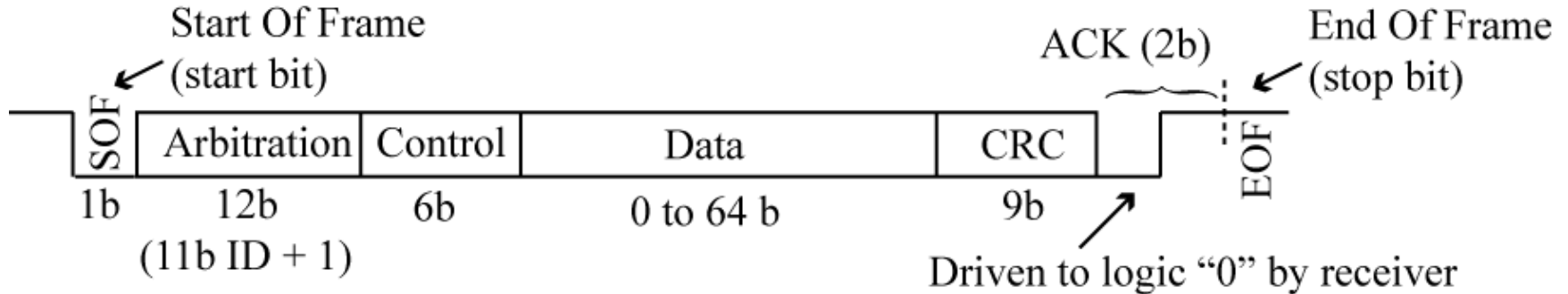
I.E. The engine processor periodically sends out information about oil temperature, engine RPMs, etc.

The Anti-Lock-Brake (ABS) system will periodically transmit information about each wheel speed.

Other systems in the car monitor the bus for these messages, and elect whether or not to receive this information.

In this way, a processor does not have to know ahead of time which other processors may be interested in its data!

A CAN Message

Start Of Frame (start bit)

ACK (2b)

End Of Frame (stop bit)

SOF	Arbitration	Control	Data	CRC		EOF	

1b   12b   6b   0 to 64 b   9b

(11b ID + 1)

Driven to logic "0" by receiver

Data packet can contain up to 64 bits. All data in all fields sent most significant bit first.

Arbitration field contains the message identifier.

Control field includes the message length.

CRC (Cyclic Redundancy Check) field is used to detect a transmission error in the message. It is a checksum that is computed based upon the Arbitration, Control, and Data fields.

Definitions: Multi-Master and Arbitration

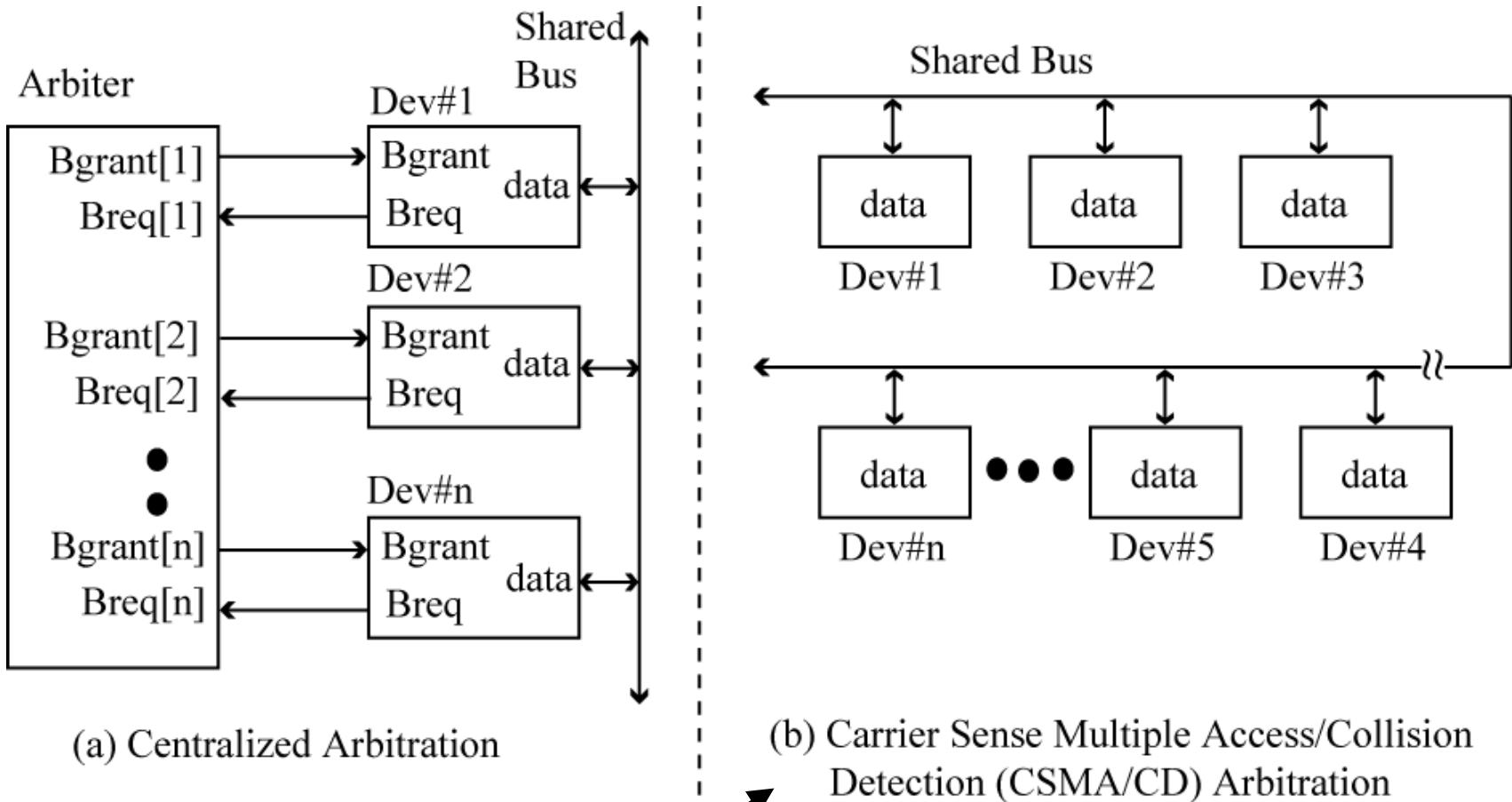A ***Multi-Master*** bus allows any processor on the bus to initiate a transaction.

The CAN is multi-master.

In a multi-master bus, must have some ***arbitration*** method when multiple processors attempt to access the bus simultaneously.

The arbitration method selects a WINNER that will be allowed to transmit its message.

Arbitration Methods

(a) Centralized Arbitration

(b) Carrier Sense Multiple Access/Collision Detection (CSMA/CD) Arbitration

CAN uses a form of CSMA/CD.

Arbitration in CAN

On the CAN bus, a logic '0' is DOMINANT, a logic '1' is RECESSIVE. This means that a logic '0' overwrites a logic '1' if both transmitted simultaneously.

When two transmitters access the bus simultaneously, both begin sending the message ID.

Both transmitters LISTEN to the bus and compare the received data against what they are transmitting. If there is a DIFFERENCE, then the processor stops sending.
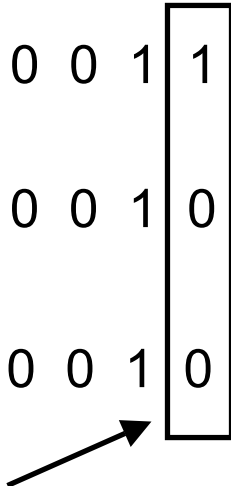
The message ID with the *lower numerical value* wins.

Arbitration in CAN (cont)

CPU #0 sends message with ID: 235 (0x0EB = 00011101011)

CPU #1 sends message with ID: 136 (0x088) = 00010001000)

CPU #0 transmits: 0 0 0 1 | 1 |

CPU #1 transmits: 0 0 0 1 | 0 | 0 0 1 0 0 0

CAN bus contains: 0 0 0 1 | 0 |

CPU #0 detects difference in what it sent and what CAN bus contains, stops sending. CPU #1 wins the arbitration, NO TIME IS LOST! Messages IDs with lower numerical value have higher priority.

Data Rates in CAN

All processors must agree on the bit signaling interval.

Bit data rate is 1/ (bit signaling interval).

Maximum data rate is dependent upon electronics used to drive signals.

Upper data rate is typically 1 Mbs (1 Million bits per second).

Clock Synchronization in CAN

Problem: Each CPU has its own clock for measuring time.

Even though each CPU is using a clock of the same frequency, the clocks can be out of phase (clock edges are not aligned), and have minor frequency differences.

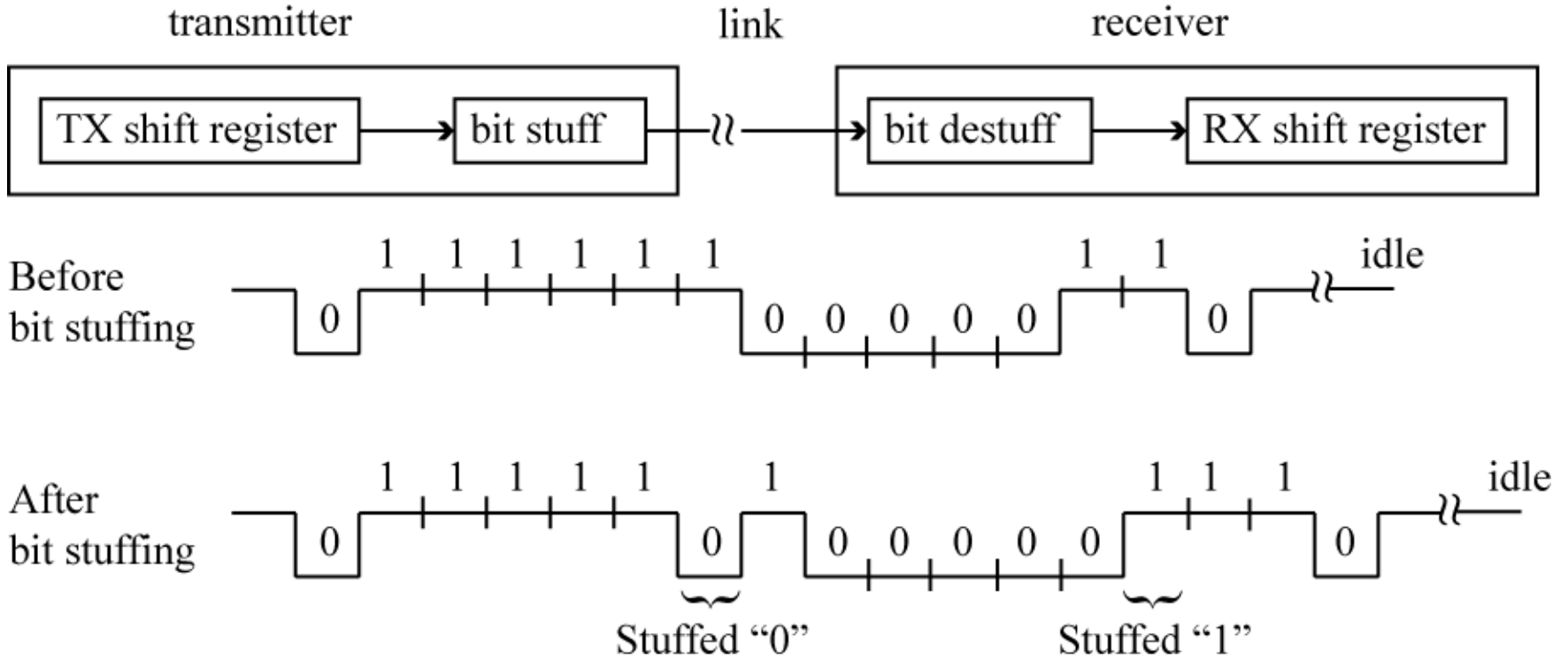How can we correct these clocks (i.e, SYNCHRONIZE the clocks?).

Use a technique known as ***BIT STUFFING*** along with a ***Phase-Locked Loop*** to synchronize clocks.

Phase Locked Loop (PLL)
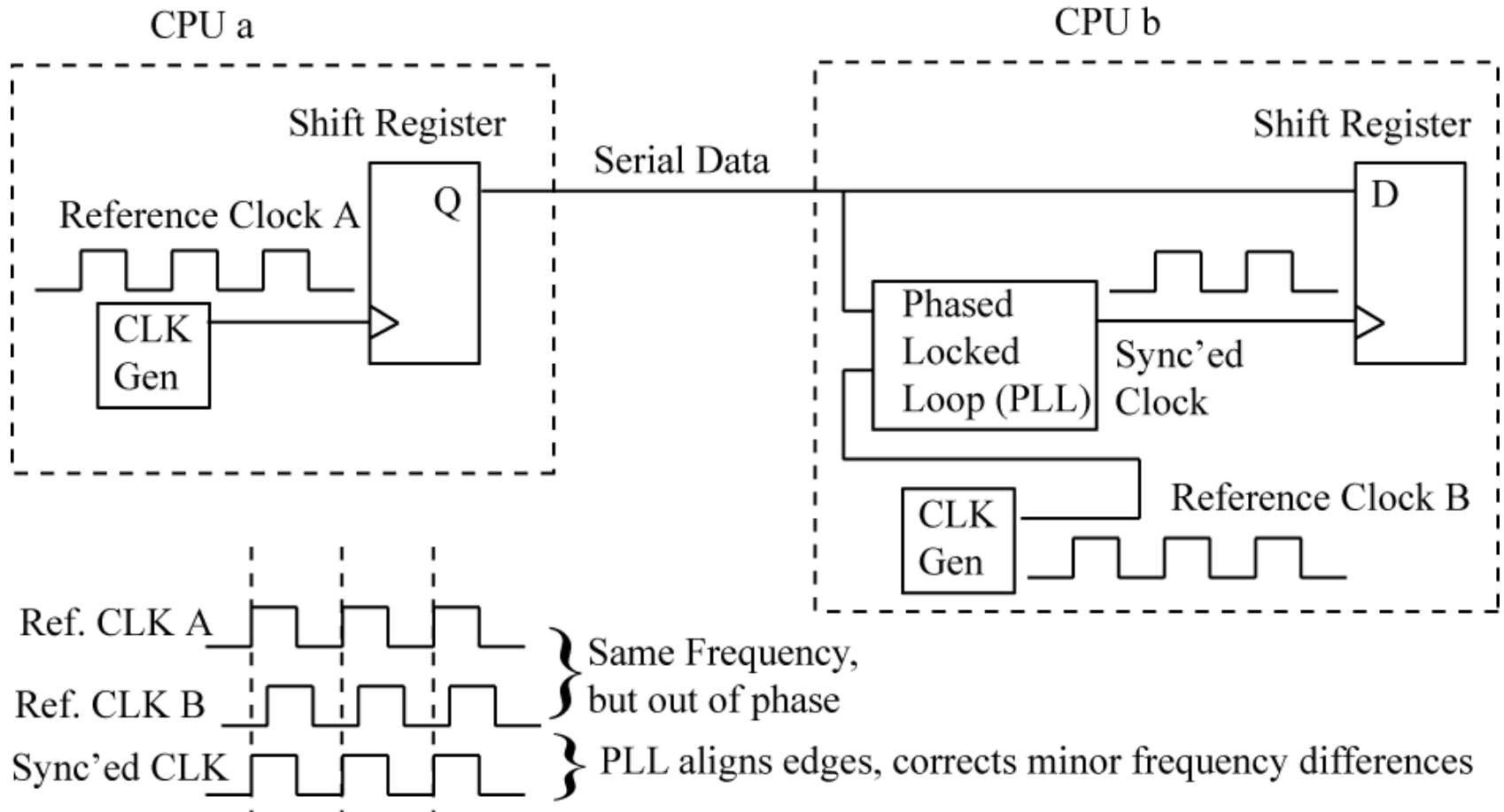
For a PLL to work correctly, the input data must have a guaranteed ***TRANSITION density***, i.e., must change voltage states within a minimum time interval.

CAN Bit Stuffing

If transmitter detects FIVE consecutive bits of the same value in bitstream, then complement is STUFFED into bit stream.

Keeps PLL synchronized to bit stream, aids in error detection.

Receiver DESTUFFs the bit stream.

Review of CAN Bus Characteristics

- Serial – data sent one bit at a time
  - Minimizes cabling size
- Half-duplex – data cannot be exchanged simultaneously between CPUs
  - Minimizes cabling size
- Differential signaling – two wires used to signal a logic '1', logic '0'
  - Improves noise rejection
- Multi-master bus, arbitration is CSMA/CD
  - Do not know number of processors on bus ahead of time, so must use CSMA/CD.
  - Message ID with lower numerical value wins

Review of CAN Bus Characteristics (cont).

- Addressing assigns ID to each message (2048 possible IDs), does not assign an ID to each node.
    - Each node determines which messages it wants to receive
    - Allows broadcast of sensor data to multiple recipients
- CAN Messages can contain up to 64 bits (8 data bytes)
    - CRC field used for error detection.
- Bit stuffing and PLLs used for clock synchronization