# Chapter 5-2: Multiprocessor Architectures

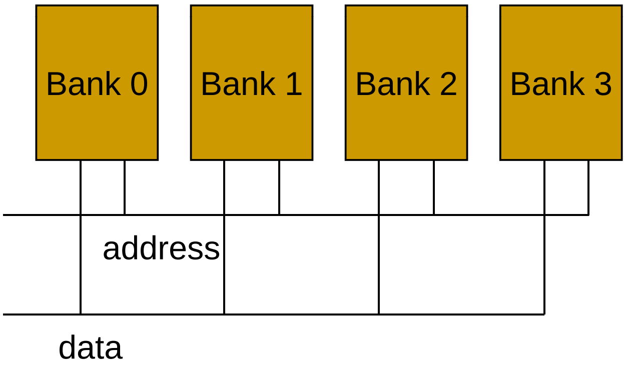
Soo-Ik Chae

## Topics

- Memory systems.
- Physically distributed multiprocessors.
- Design methodologies.

# Parallel memory systems

- n memory banks can be accessed independently.
- Peak access rate given by n parallel accesses.
    - Can be achieved If the access pattern is properly laid out like (0,1,2,3,0,1,2,3,…)



Bank 0

Bank 1

Bank 2

Bank 3

address

data

# Parallel memory systems

- Performance can be estimated statistically
    - $\lambda$: probability of non-sequential memory access.
    - The probability of a string of k sequential accesses is $p(k)=\lambda(1-\lambda)^{k-1}$
    - Mean length of a sequential access sequence is $L_b$

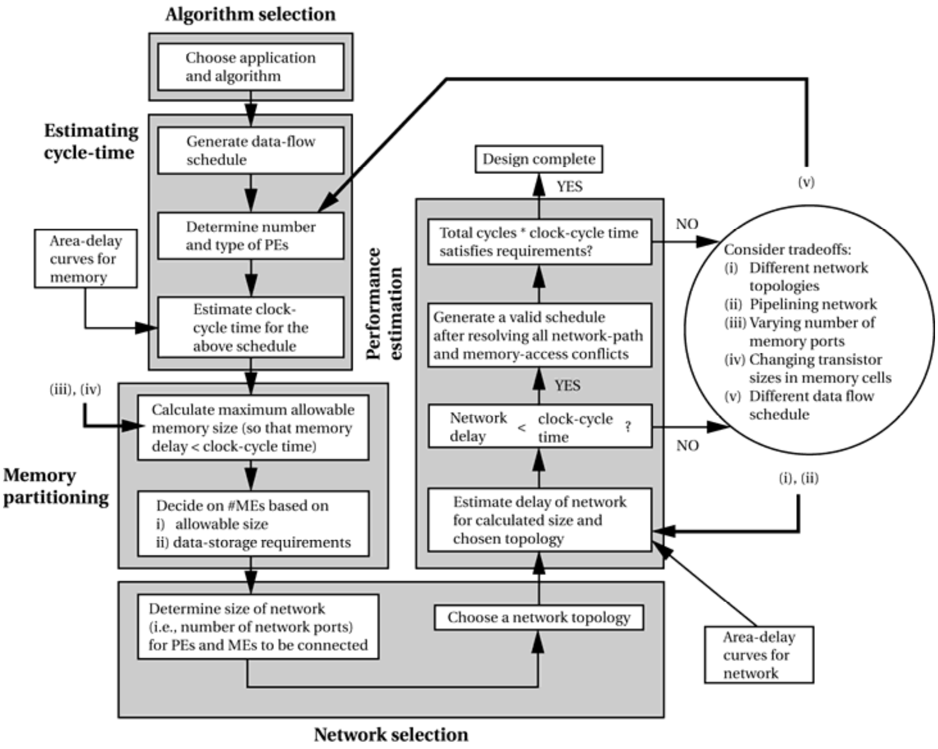
$$L_b = \sum_{1 \le k \le \infty} kp(k) = \frac{1-(1-\lambda)^m}{\lambda}.$$

- We can use program statistics to estimate the average probability of non-sequential accesses, and design the memory system accordingly
- We can also use software techniques to maximize the length of access sequences whenever possible.

# Memory system design

- Models for memory
  - Model parameters: area, performance, energy.
- Delay is a nonlinear function of memory size.
  - Wire delay ~ between O(n) and O($n^2$)
- Delay and energy are a nonlinear function of the number of ports.

# Memory system design methodology


Algorithm selection
Choose application and algorithm
Estimating cycle-time
Generate data-flow schedule
Determine number and type of PEs
Area-delay curves for memory
Estimate clock-cycle time for the above schedule
(iii), (iv)
Calculate maximum allowable memory size (so that memory delay < clock-cycle time)
Memory partitioning
Decide on #MEs based on
i) allowable size
ii) data-storage requirements
Determine size of network (i.e., number of network ports) for PEs and MEs to be connected
Choose a network topology
Network selection
Performance estimation
Design complete
YES
Total cycles * clock-cycle time satisfies requirements?
NO
Generate a valid schedule after resolving all network-path and memory-access conflicts
YES
Network delay < clock-cycle time ?
NO
Estimate delay of network for calculated size and chosen topology
(v)
Consider tradeoffs:
(i) Different network topologies
(ii) Pipelining network
(iii) Varying number of memory ports
(iv) Changing transistor sizes in memory cells
(v) Different data flow schedule
(i), (ii)
Area-delay curves for network

# Memory system design

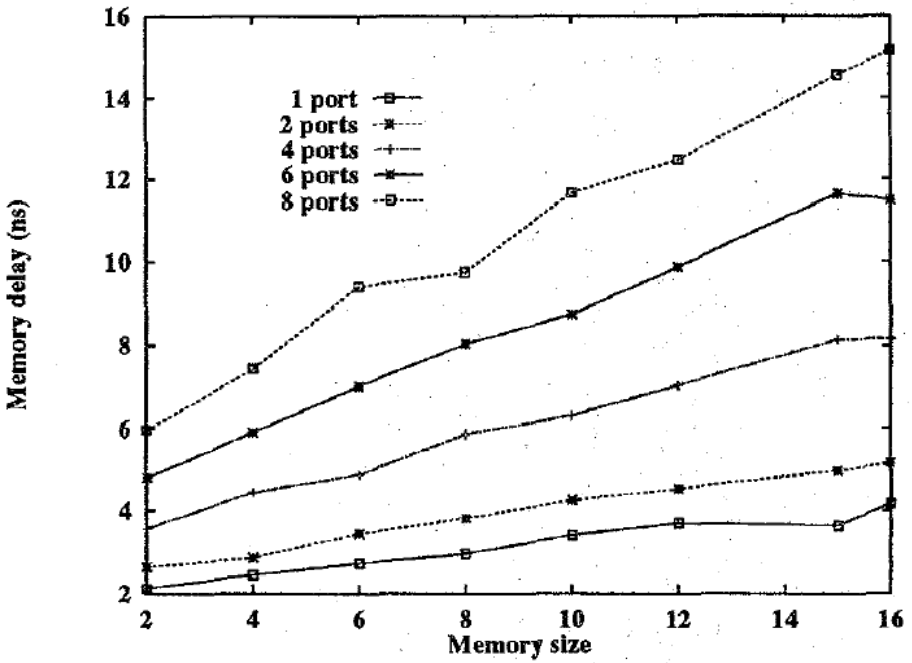
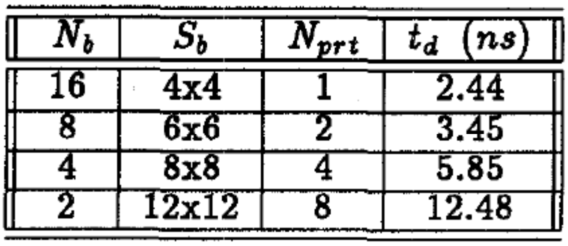


Figure 4: Delay variation with number of ports

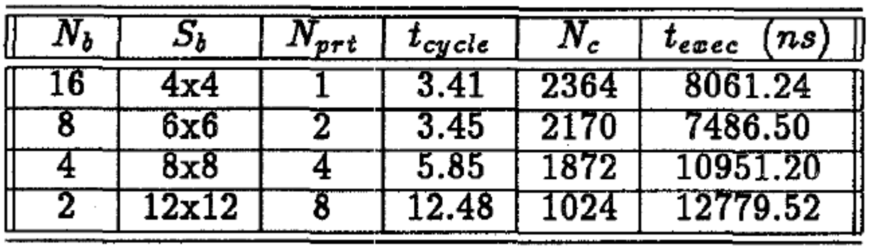
# Memory system design

Table 2: Delay of minimum-sized memory banks



| $N_b$ | $S_b$ | $N_{prt}$ | $t_d$ $(ns)$ |
|---|---|---|---|
| 16 | 4x4 | 1 | 2.44 |
| 8 | 6x6 | 2 | 3.45 |
| 4 | 8x8 | 4 | 5.85 |
| 2 | 12x12 | 8 | 12.48 |

Table 3: Fixed datapath delay (small transistors)



| $N_b$ | $S_b$ | $N_{prt}$ | $t_{cycle}$ | $N_c$ | $t_{exec}$ $(ns)$ |
|---|---|---|---|---|---|
| 16 | 4x4 | 1 | 3.41 | 2364 | 8061.24 |
| 8 | 6x6 | 2 | 3.45 | 2170 | 7486.50 |
| 4 | 8x8 | 4 | 5.85 | 1872 | 10951.20 |
| 2 | 12x12 | 8 | 12.48 | 1024 | 12779.52 |

# Heterogeneous memory systems

- Heterogeneous memory improves real-time performance:
  - Accesses to the same bank interfere, even if not to the same location.
  - Segregating real-time locations improves predictability, reduces access time variance.
- Heterogeneous memory improves power:
  - Smaller blocks with fewer ports consume less energy.

$$E_M = \sum_{i \in modules} E_{i,module} + \sum_{j \in switch} E_{j,switch} + \sum_{k \in wires} E_{k,wire}.$$

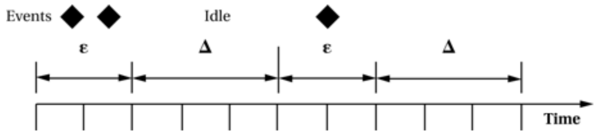
# Networks and physically-distributed embedded systems

- Examples: automobiles, airplanes.
- Nodes connected by a network.
  - Network delay is noticeable.
- Reasons for physically distributed nodes:
  - Must keep some computation close to mechanics to reduce latency.
  - May reduce network bandwidth by processing data locally.
  - Modular design may be assembled from components by different vendors.
  - Fault tolerance into systems

# Time-Triggered Architecture

- TTA is a distributed architecture for real-time control.
- TTA has a notion of real time.
  - Correct partial order is not sufficient.
- TTH timestamp is based on GPS clock.
  - 64-bit value.
  - Fractions of second in three lower bytes, seconds in five upper bytes.
  - GPS epoch starts at 0:00:00 UCT Jan 6, 1980.

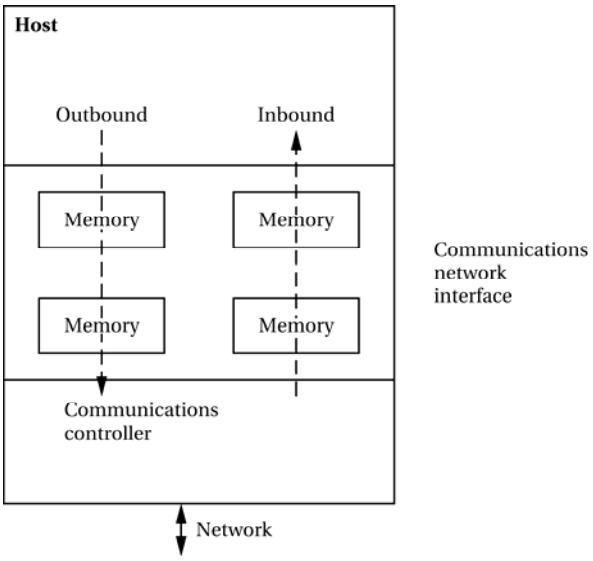
# Sparse model of time



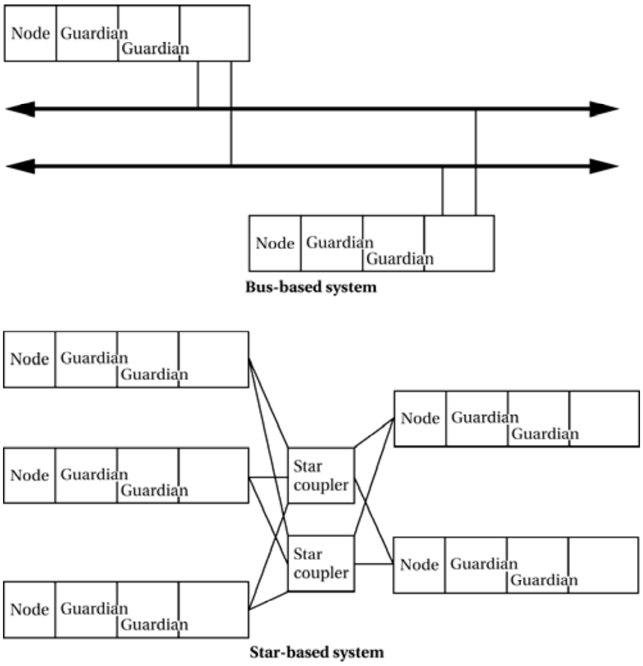
- Allows predictable interaction between physical time and discrete time.
- Active periods denoted by ε.
- Idle periods denoted by δ.
- Events occur during ε, never during δ.
- Duration of ε, δ is larger than precision of the clock.
- The sparse time model ensures that events will not be reordered due to small variations in the clock between nodes.
  - In a dense timing model, an arriving event may be stamped with two different times due to variations in the clock value at different nodes.

# Communications network interface

- CNI Helps maintain consistent view of time.
  - Between host controller and communications controller.
- Enforces unidirectional flow of data.
  - One inbound, one outbound channel.
- Buffering ensures that tasks on the host are not delayed by unpredictable communication delays



# TTA topologies



# Cliques

- In a fault-tolerant system, failures cause internal inconsistencies.
  - Different nodes have different views of the system state.
- Clique avoidance algorithm identifies faulty nodes.
  - Protocols can identify state inconsistency.
  - Action on faulty nodes is determined by the application.

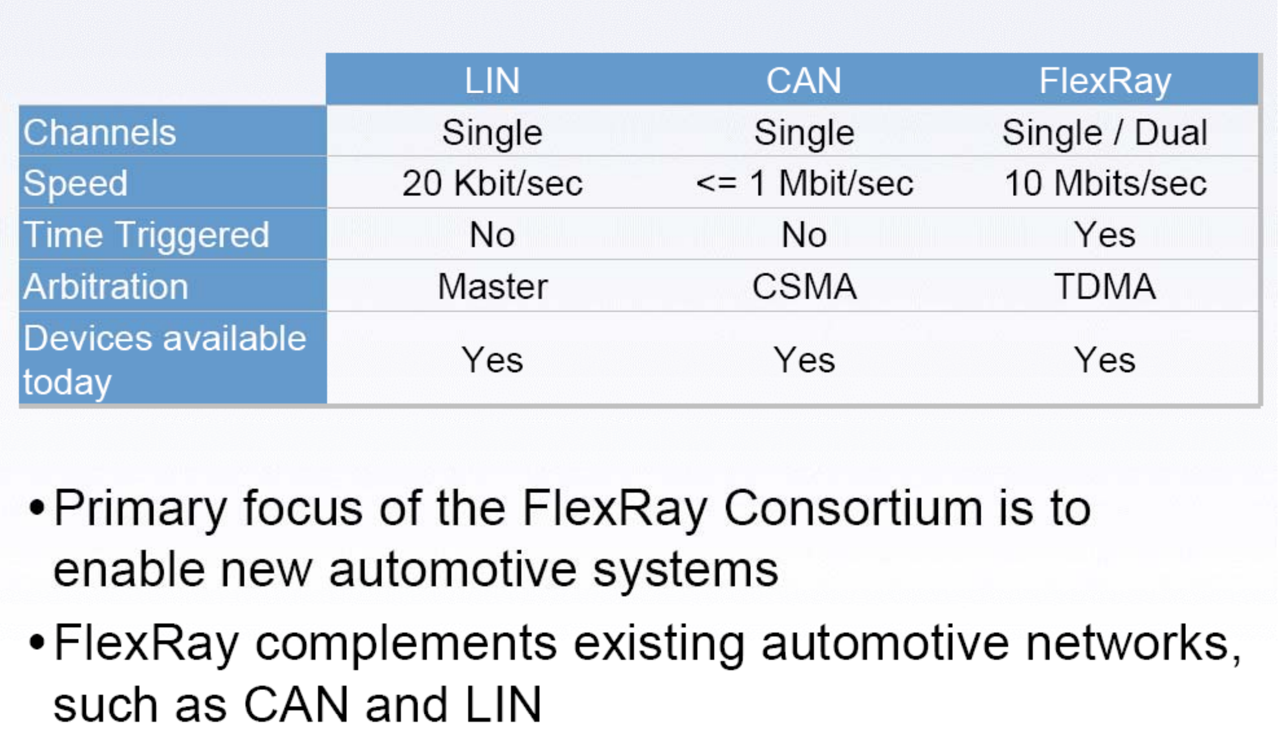
# Typical Protocols

| 속도 | 프로토콜 | 특성 |
|---|---|---|
| 저속 | LIN | 보통 20 kbps<br>스마트 센서와 액츄에이터 같은 단순한 온-오프 기기의 통신 |
| 중 • 저속 | CAN | 최대 1 Mbps (High-speed CAN) ECU 간의 통신. 제어 신호<br>제어기용 네트워크 표준으로 정립 |
| 고속 | MOST | 현재 25 Mbps. MOST2는 150Mbps. 멀티미디어 버스<br>유럽을 중심으로 양산 중 |
|  | IEEE 1394 | 100 Mbps 이상. 멀티미디어 버스<br>미국을 중심으로 표준화 진행 중 |

# FlexRay

- Shortcomings of existing solutions
  - Data rate
  - Deterministic behavior
  - Fault tolerance
  - Topology
- Sept 2000
  - Foundation of the FlexRay Consortium
  - BMW, DaimlerChrysler, Philips, Motorola

# FlexRay

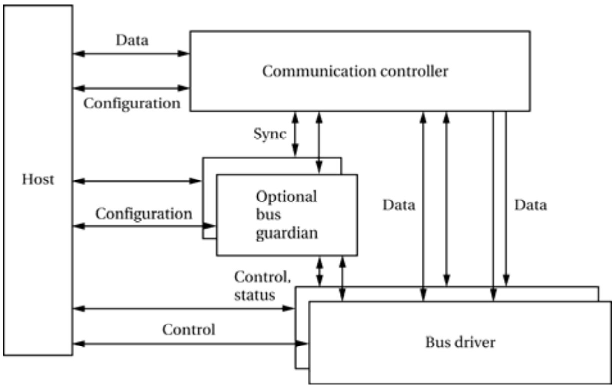


| | LIN | CAN | FlexRay |
|---|---|---|---|
| Channels | Single | Single | Single / Dual |
| Speed | 20 Kbit/sec | <= 1 Mbit/sec | 10 Mbits/sec |
| Time Triggered | No | No | Yes |
| Arbitration | Master | CSMA | TDMA |
| Devices available today | Yes | Yes | Yes |

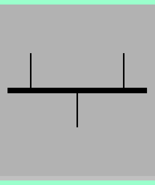
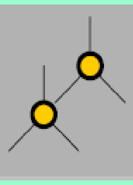
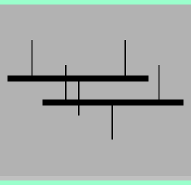
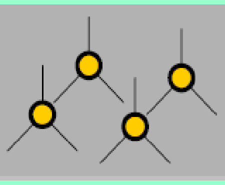
- Primary focus of the FlexRay Consortium is to enable new automotive systems
- FlexRay complements existing automotive networks, such as CAN and LIN

# FlexRay

- Host runs applications.
- Communication controller provides high-level functions.
  - Interface to host
  - Message processing: transmission, reception
  - Clock synchronization
- Bus guardians watch system for errors.



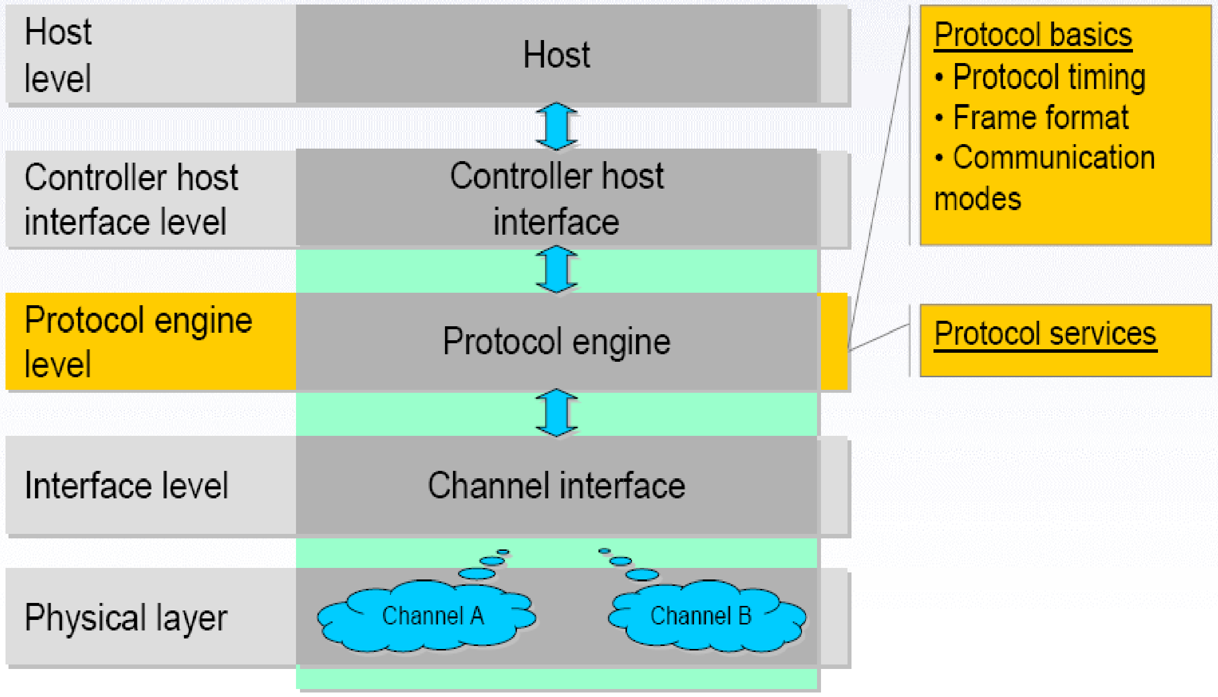
# Network topology overview

| | Bus | Multiple star | |
|---|---|---|---|
| Single channel |  |  | reduced wire-harness, experience, cost |
| Dual channel |  |  | tolerates one faulty channel |
| | passive medium, most experience, cost efficient | allows for high data rates, increases error containment | Electrical & optical physical layer |

# Interface level overview

- FlexRay supports bus guardian at physical interface
  - enforces error containment in the time domain
  - performs error detection in the time domain
- Bus guardian interacts with
  - communication controller
    - signal monitoring
    - synchronization
  - host processor
    - configuration
    - activation / deactivation
    - error signalling

# FlexRay architecture levels – Level 3

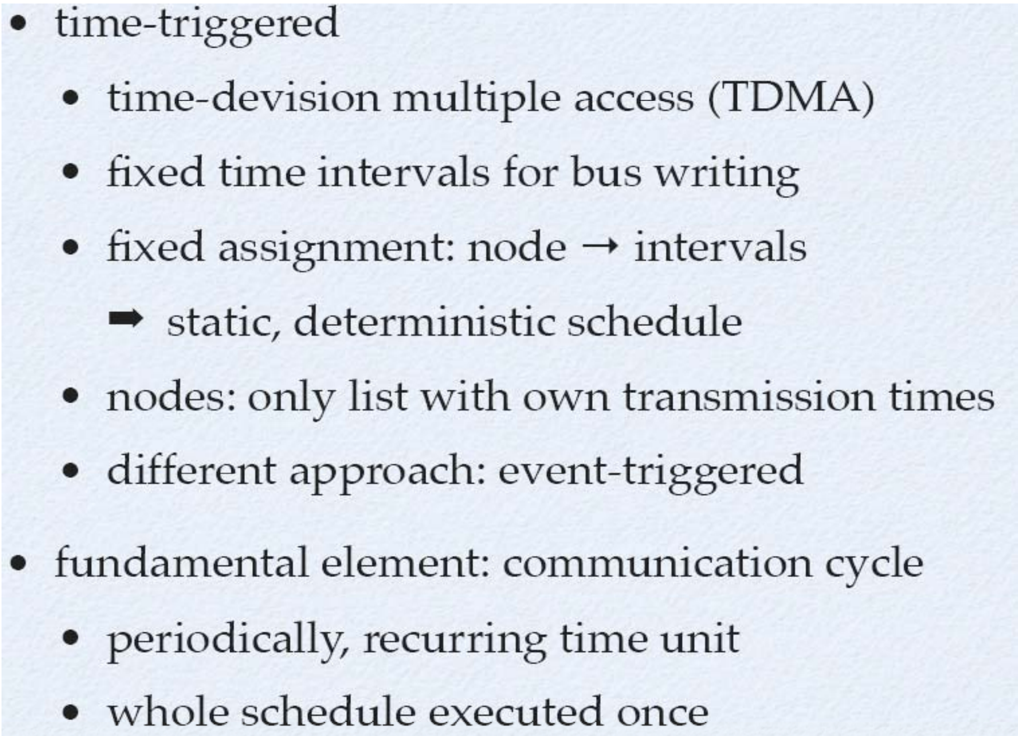


| | |
|---|---|
| Host level | Host |
| Controller host interface level | Controller host interface |
| Protocol engine level | Protocol engine |
| Interface level | Channel interface |
| Physical layer | Channel A<br>Channel B |

Protocol basics
- Protocol timing
- Frame format
- Communication modes

Protocol services

# FlexRay schedule



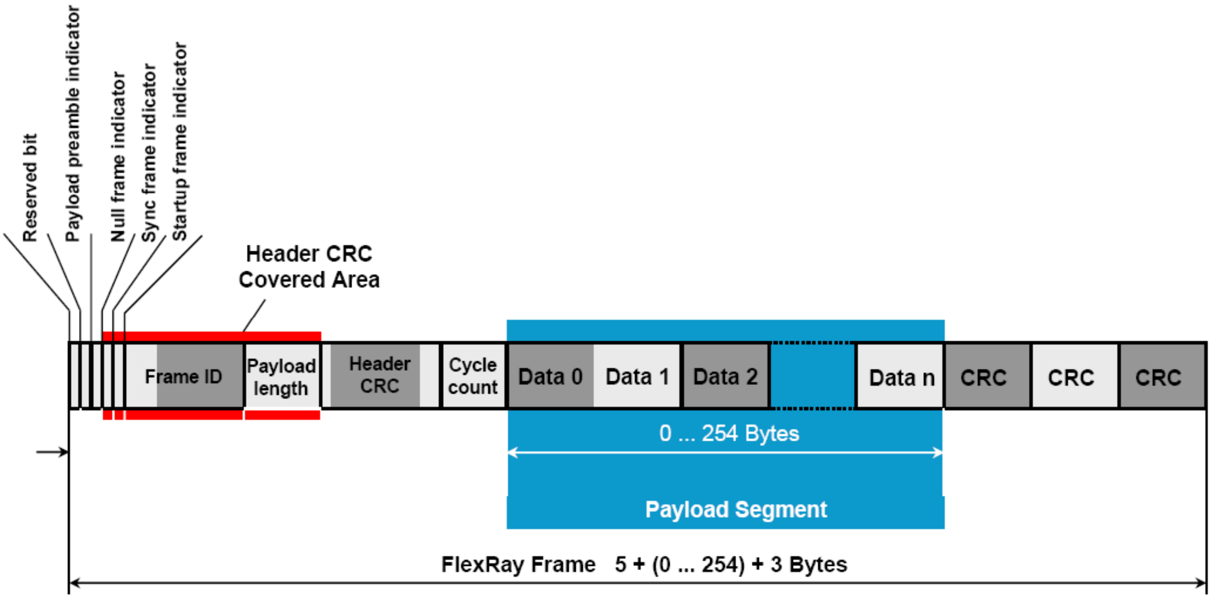
- time-triggered
  - time-devision multiple access (TDMA)
  - fixed time intervals for bus writing
  - fixed assignment: node → intervals

    ➡ static, deterministic schedule
  - nodes: only list with own transmission times
  - different approach: event-triggered
- fundamental element: communication cycle
  - periodically, recurring time unit
  - whole schedule executed once

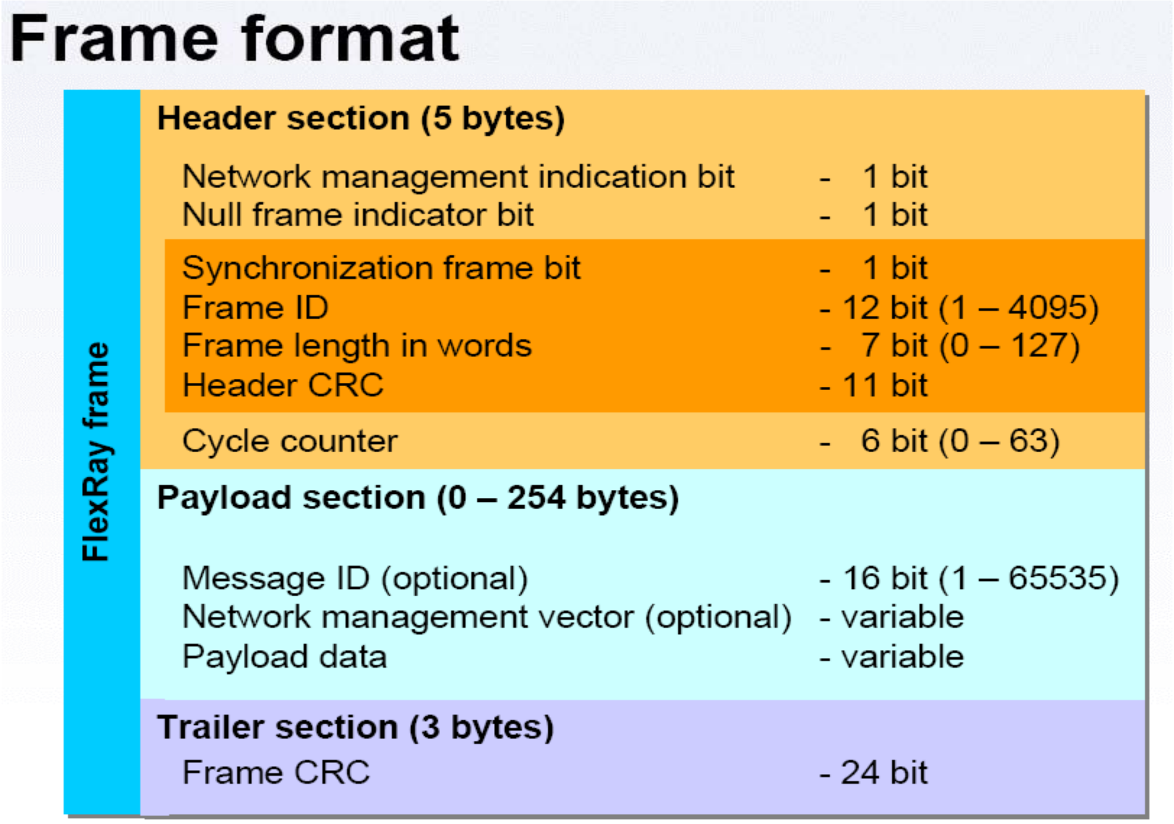
# FlexRay timing

- Action points are boundaries between macroticks.
- Arbitration grid determines boundaries between messages.
- Communication cycle:
  - Static segment.
  - Dynamic segment.
  - Symbol window.
  - Network idle time.

# FlexRay frame format



- 254 bytes, 8 Bytes overhead (5 header incl. header CRC, 3 frame CRC) plus start/stop bits



# Frame format

FlexRay frame

**Header section (5 bytes)**

| | |
|---|---|
| Network management indication bit | - 1 bit |
| Null frame indicator bit | - 1 bit |
| Synchronization frame bit | - 1 bit |
| Frame ID | - 12 bit (1 – 4095) |
| Frame length in words | - 7 bit (0 – 127) |
| Header CRC | - 11 bit |
| Cycle counter | - 6 bit (0 – 63) |

**Payload section (0 – 254 bytes)**

| | |
|---|---|
| Message ID (optional) | - 16 bit (1 – 65535) |
| Network management vector (optional) | - variable |
| Payload data | - variable |

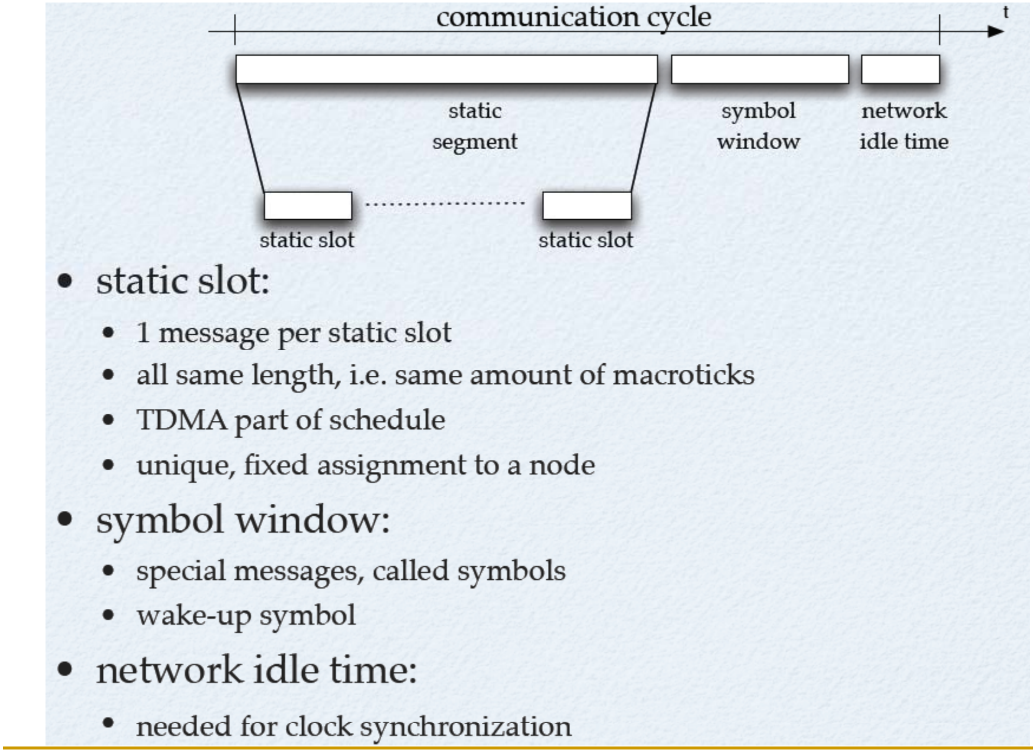
**Trailer section (3 bytes)**

| | |
|---|---|
| Frame CRC | - 24 bit |

# FlexRay real-time performance

- Static phase is scheduled statically for real-time behavior.
- Dynamic phase provides non-time-critical time slots.
- Microtick comes from application internal clock.
- Macrotick comes from clusterwide synchronized clock.
- Creates a temporal firewall between time-sensitive and no-time-sensitive transmissions

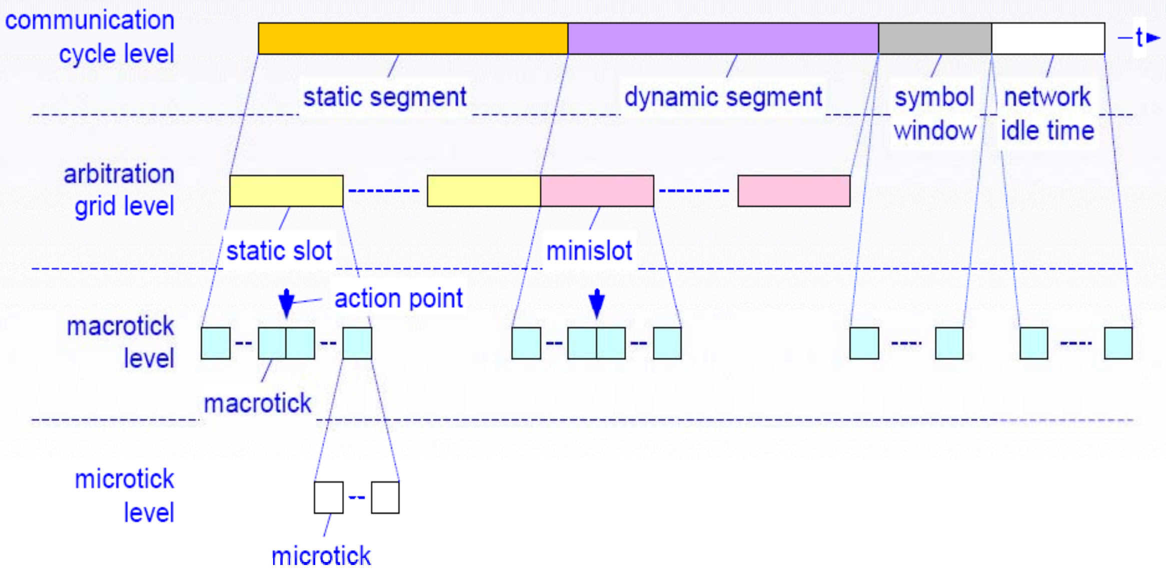
# Communication cycle



- static slot:
  - 1 message per static slot
  - all same length, i.e. same amount of macroticks
  - TDMA part of schedule
  - unique, fixed assignment to a node
- symbol window:
  - special messages, called symbols
  - wake-up symbol
- network idle time:
  - needed for clock synchronization

# Protocol timing

Protocol timing related to the schedule of the communication cycle



# FlexRay Encoding Approach

- **Data sent as NRZ bytes**
  - TSS = Transmit Start Sequence (LOW for 5-15 bits)
  - FSS = Frame Start Sequence (one HI bit)
  - BSS = Byte Start Sequence (similar to start/stop bits in other NRZ)
  - FES = Frame End Sequence (END symbol for frame – LO + HI)

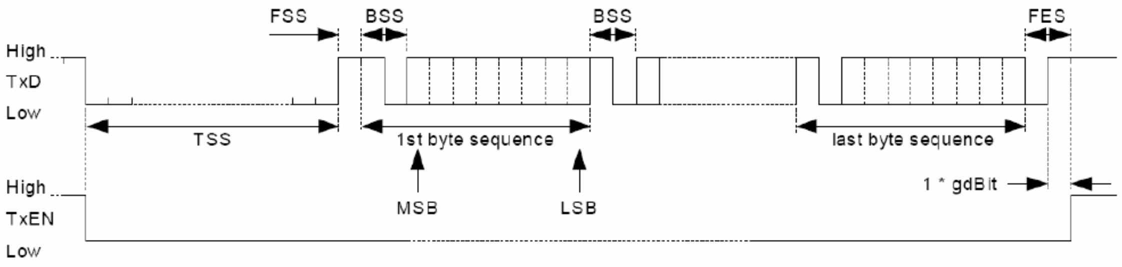
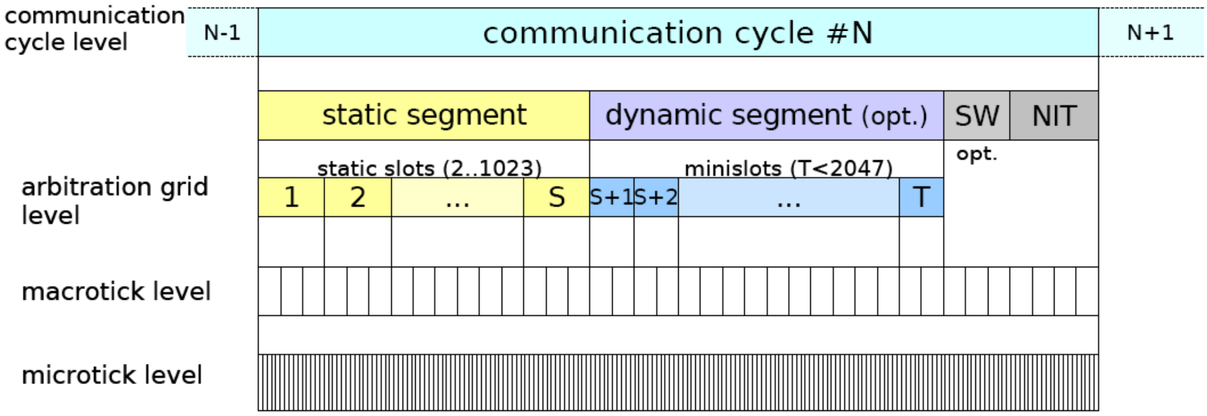


**Figure 3-2: Frame encoding in the static segment.** [FlexRay04]

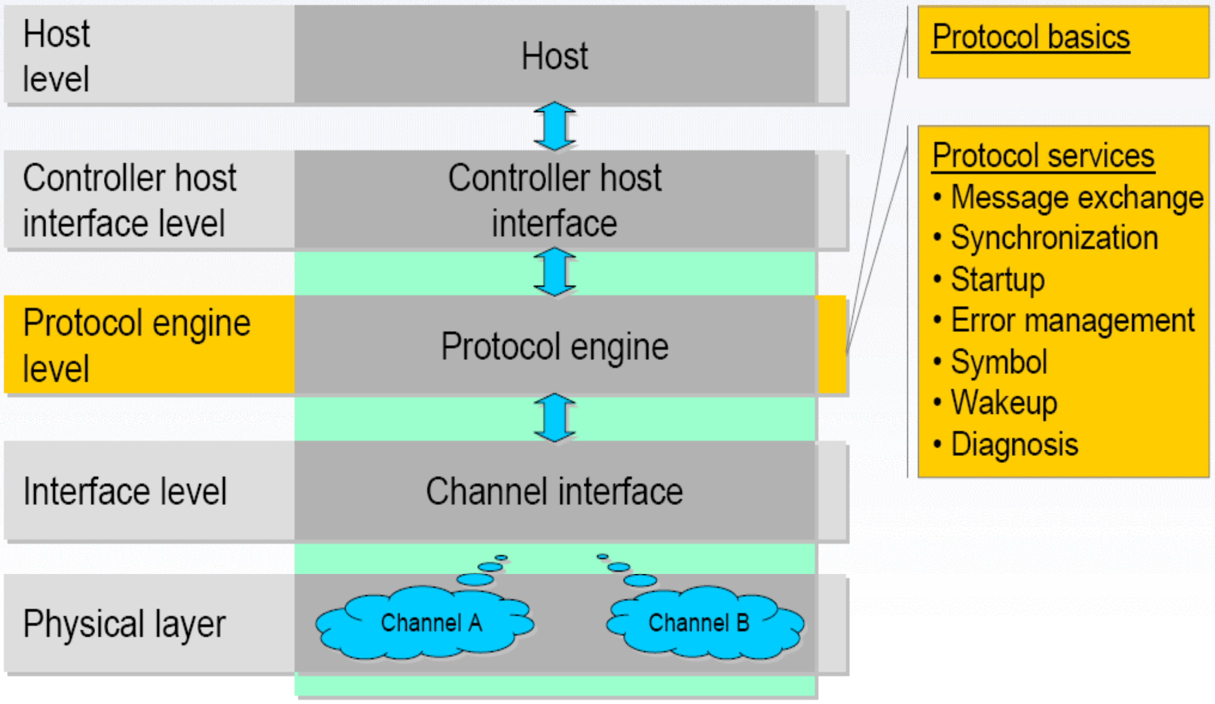
- **Dynamic segment frames are similar**
  - Adds a DTS = dynamic trailing sequence field; helps line up minislots

# Communication cycle

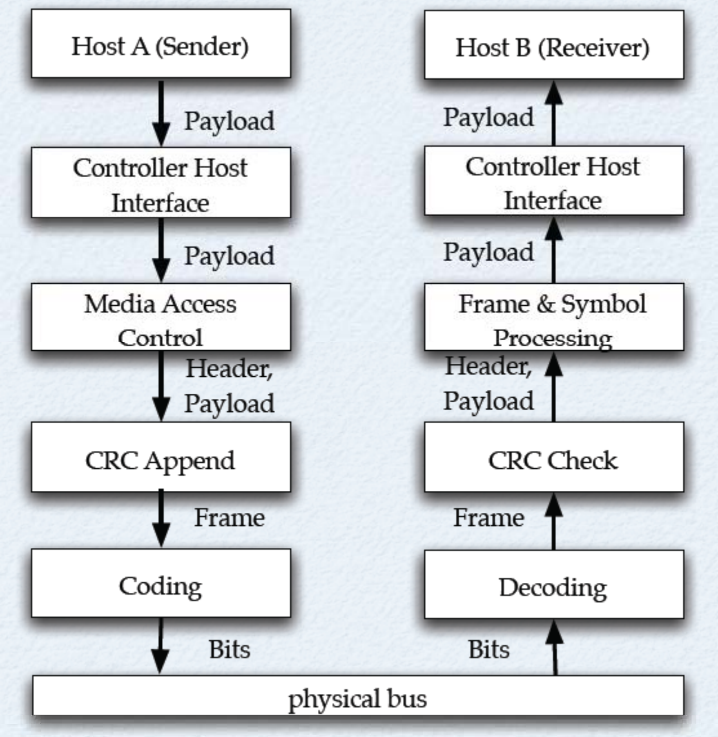
- Maximum duration: 16000µs
- Static segment: TDMA
- Dynamic Segment: minislotting scheme
- SW (symbol window), NIT (network idle time)



# FlexRay architecture levels – Level 3

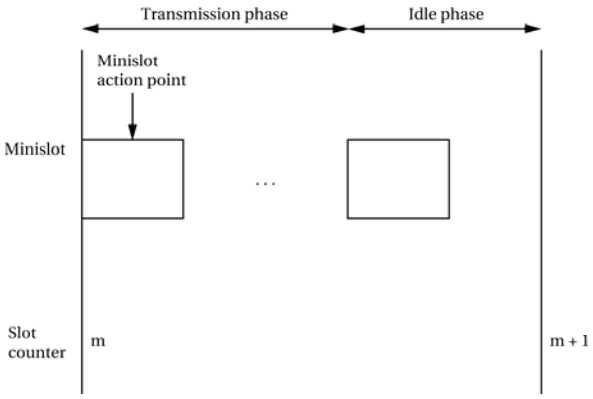


# Message Processing

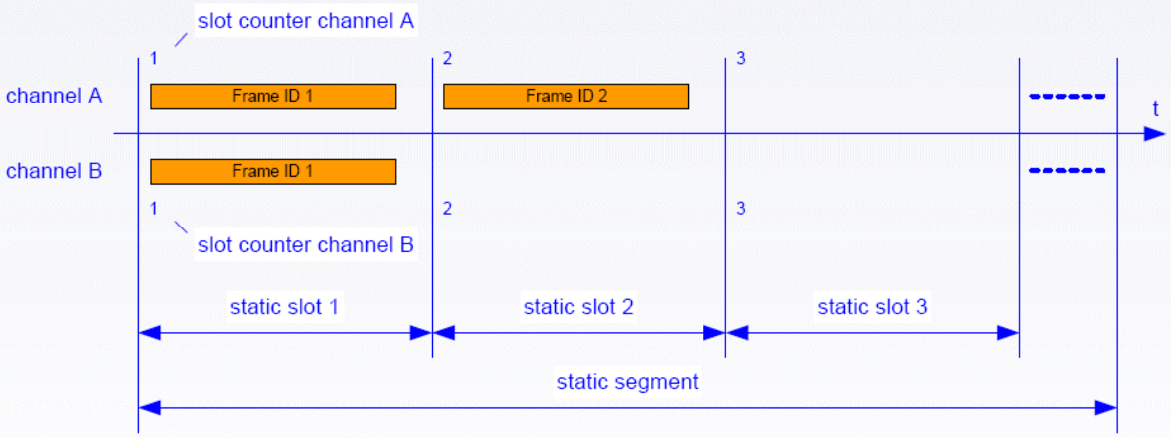


# FlexRay segment timing

- Slots are arbitrated using a deterministic algorithm.
- Messages sent at minislot boundaries.
- Message lasts longer than a minislot if sent.



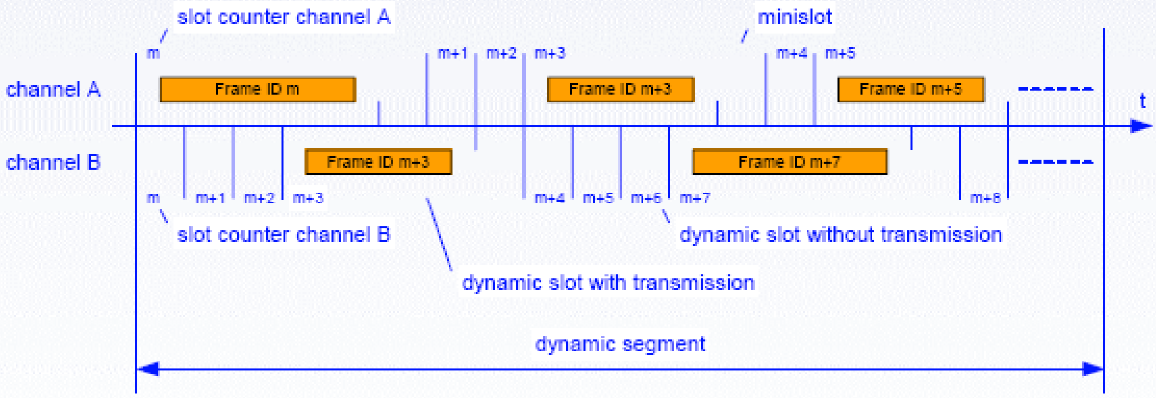
# Statically scheduled frames



- Frames of static length assigned uniquely to slots of static duration
  - Frame sent when assigned slot matches slot counter
- Multiple slots per node assignable / message content variable
  - Decouple agreement/diagnosis cycles from communication cycle
  - Support intra-cycle application level agreement
- BG protection of static slots

page

# Dynamically scheduled frames



- Dynamic bandwidth allocation
  - per node as well as per channel
- Collision-free arbitration via unique IDs and minislot counting
  - Frame sent when scheduled frame ID matches slot counter
- No BG protection within dynamic segment

page

# Clock Synchronization

- Clock synchronization component
  - Perform two ways of time synchronization
  - Generate macroticks, a local time unit.
- Macrotick
  - Their length depends on the local oscillator clock
- The aim is that a macrotick has approximately the same length on any bus controller within the cluster relative to real time.
- That is a non-trivial problem in distributed systems because the local oscillator clocks may drift due to fabrication tolerances or environment differences.
- For any kind of correction a time difference between the local view of time and the view of other bus controllers is needed.
- This can be achieved by so called sync messages.

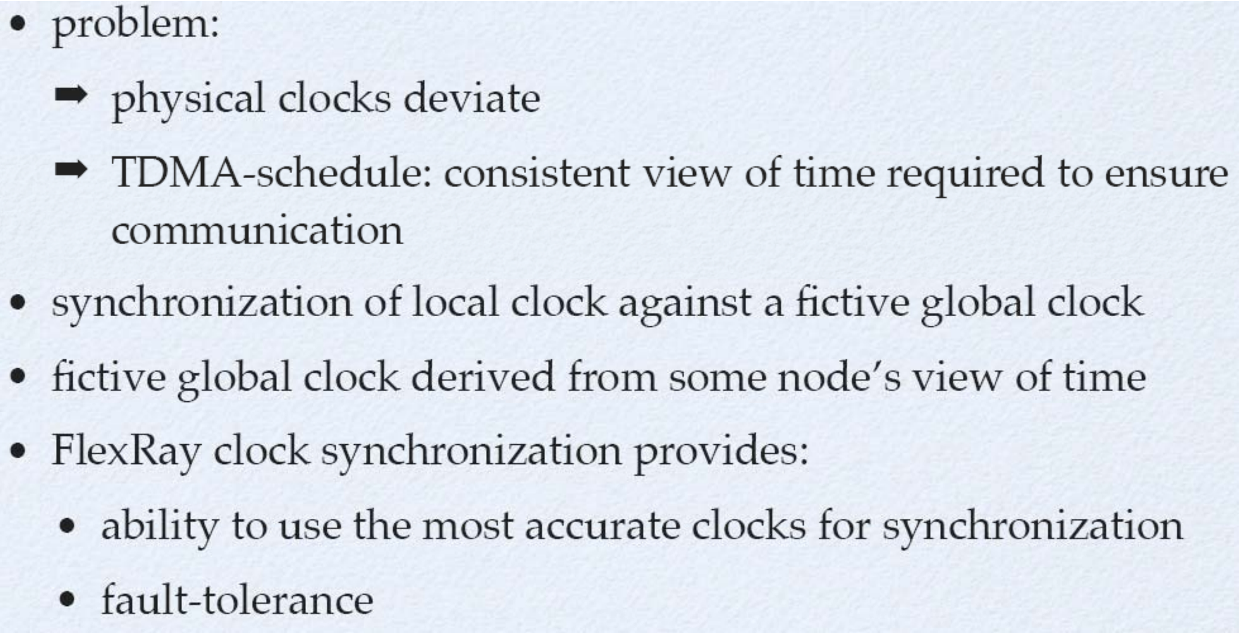
# Clock Synchronization

- If a bus controller receives a sync message, the difference between expected and observed arrival time calculated and stored.
- During a communication cycle (round), up to 15 sync messages are received.
- The clock synchronization FlexRay performs
  - Offset correction
  - Rate correction

# Clock Synchronization

- Offset correction: a fault-tolerant midpoint algorithm computes an average over the time differences from one communication cycle and the next schedule execution is delayed or starts earlier
- Rate correction: minimize the difference between time views by adjusting the length of a macrotick.
  - Computes the difference between the expected and observed arrival time of a sync message in two consecutive communication rounds
  - Then the macrotick length is adjusted accordingly.

# Clock Synchronization



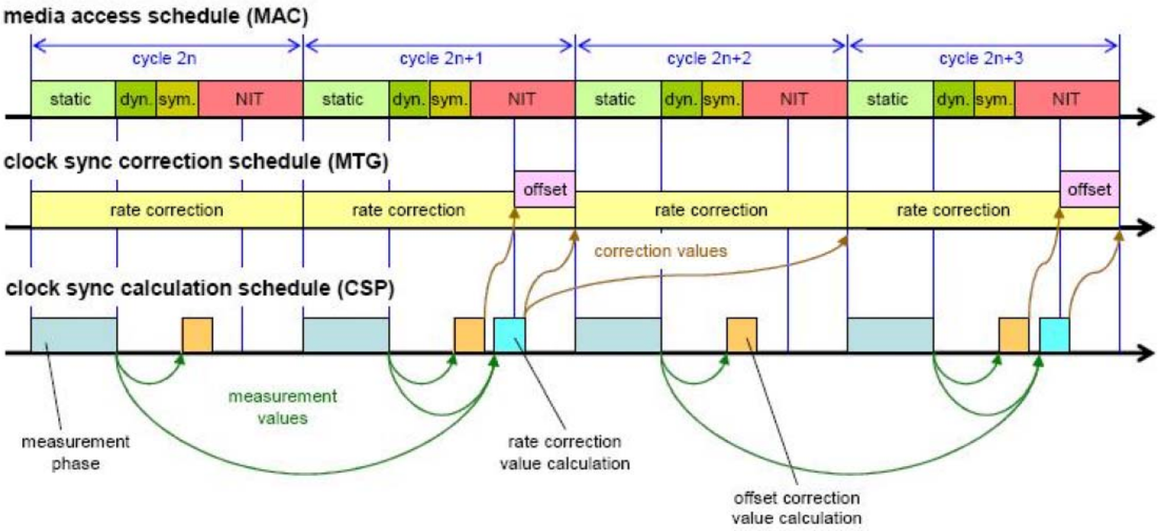
- problem:
  - ➡ physical clocks deviate
  - ➡ TDMA-schedule: consistent view of time required to ensure communication
- synchronization of local clock against a fictive global clock
- fictive global clock derived from some node's view of time
- FlexRay clock synchronization provides:
  - ability to use the most accurate clocks for synchronization
  - fault-tolerance

# Synchronization

- Local view of the global time represented by
  - Cycle counter
  - Macrotick
  - Microtick
- 2..15 *Sync Nodes* per cluster: Sync Frame Indicator Bit in static segment
- Nodes measure time between expected and observed arrival of sync frames
- Calculate deviation from global time
  - Offset correction: Shorten/lengthen NIT
  - Rate correction: Change number of µT per cycle

# FlexRay timekeeping

- Global time is synthesized by clock synchronized process (CSP).
- Macroticks are managed by macrotick generation process. (MTG: macrotick generation process)


media access schedule (MAC)
cycle 2n
cycle 2n+1
cycle 2n+2
cycle 2n+3
static
dyn.
sym.
NIT
clock sync correction schedule (MTG)
rate correction
offset
correction values
clock sync calculation schedule (CSP)
measurement values
measurement phase
rate correction value calculation
offset correction value calculation

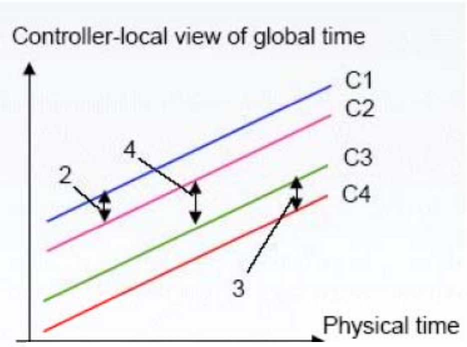
# Fault tolerant midpoint (FTM) algorithm

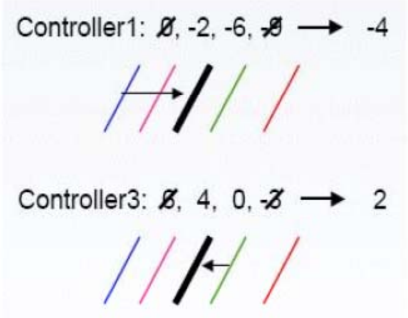
- Is used for clock state correction (Offset correction)
- First, the valid time difference is sorted and the k largest and the k smallest values are discarded.
- Then, the largest and the smallest of the remaining values are averaged for the calculation of the midpoint value which serves as the state correction term vOffsetCorection that indicates by how many microticks the node's communication cycle length should be changed.
- k= 0 for up to 2 values

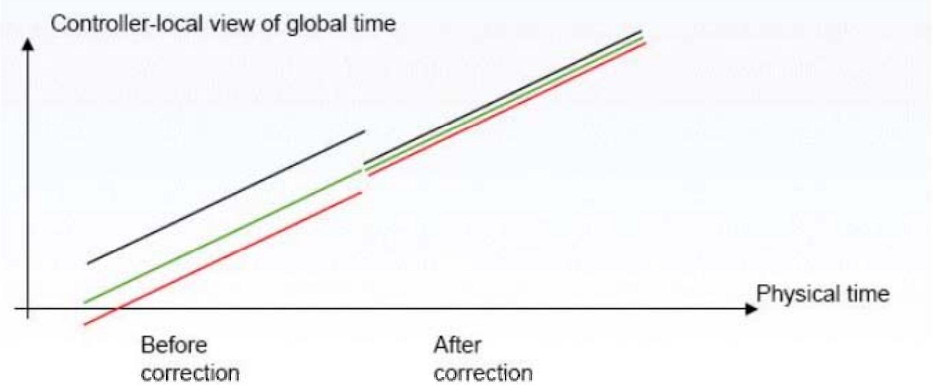
  1 for up to 7 values

  2 for more than 7 values.
- Clock state correction takes place during the network idle time NIT in every second communication cycle.

# FTM algorithm



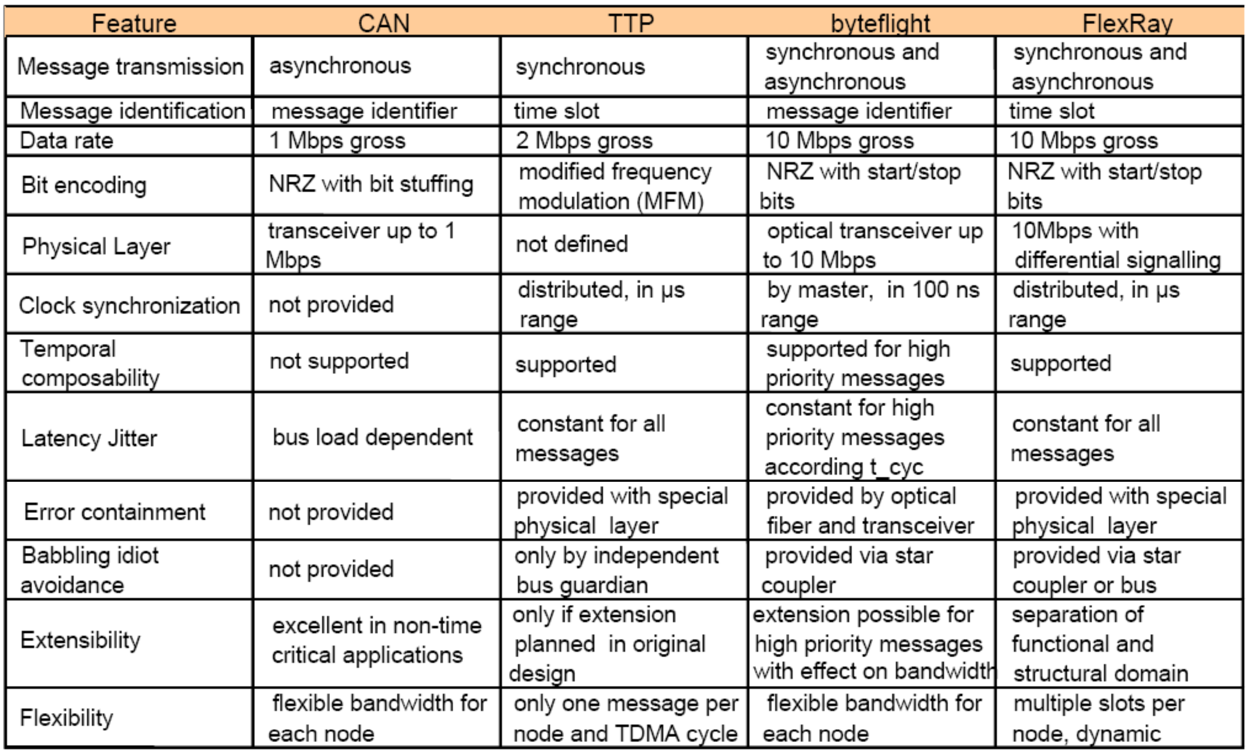

Controller1: ~~8~~, -2, -6, ~~-8~~ → -4
Controller3: ~~8~~, 4, 0, ~~-2~~ → 2



# FlexRay Summary

- Flexible and scalable bus system
- TDMA: guaranteed latency and jitter
- FTDMA: prioritizing of messages
- 10Mbit/s data rate on 2 channels
- Fault-tolerant clock synchronization
- Fault tolerance: extensive error detection and signaling, bus guardian
- Expected to be the de-facto standard for high speed automotive control applications

# Evolution of Vehicle Networks



| Feature | CAN | TTP | byteflight | FlexRay |
|---|---|---|---|---|
| Message transmission | asynchronous | synchronous | synchronous and asynchronous | synchronous and asynchronous |
| Message identification | message identifier | time slot | message identifier | time slot |
| Data rate | 1 Mbps gross | 2 Mbps gross | 10 Mbps gross | 10 Mbps gross |
| Bit encoding | NRZ with bit stuffing | modified frequency modulation (MFM) | NRZ with start/stop bits | NRZ with start/stop bits |
| Physical Layer | transceiver up to 1 Mbps | not defined | optical transceiver up to 10 Mbps | 10Mbps with differential signalling |
| Clock synchronization | not provided | distributed, in µs range | by master, in 100 ns range | distributed, in µs range |
| Temporal composability | not supported | supported | supported for high priority messages | supported |
| Latency Jitter | bus load dependent | constant for all messages | constant for high priority messages according t_cyc | constant for all messages |
| Error containment | not provided | provided with special physical layer | provided by optical fiber and transceiver | provided with special physical layer |
| Babbling idiot avoidance | not provided | only by independent bus guardian | provided via star coupler | provided via star coupler or bus |
| Extensibility | excellent in non-time critical applications | only if extension planned in original design | extension possible for high priority messages with effect on bandwidth | separation of functional and structural domain |
| Flexibility | flexible bandwidth for each node | only one message per node and TDMA cycle | flexible bandwidth for each node | multiple slots per node, dynamic |

# Aircraft networks

- Similar to automobile design, but more stringent requirements
  - More sensitive to weight
  - More complex control (3D)
- Avionics categories:
  - Instrumentation.
  - Navigation/communication.
  - Control.
- Control networks must perform hard real-time, safety-critical tasks.
- Management networks control noncritical devices.
- Passenger networks manage entertainment, internet access, etc.

# ARINC 644 standard

- Aircraft network is divided into four domains with firewalls between them:

1. Flight deck network is deterministic.
2. Separate network for OEM equipment with temporal determinism.
3. Airline systems network supports entertainment, etc.
4. Passenger subnetwork provides Internet access.

# Multiprocessor design methodologies

- MPSoC built from many hardware and software modules.
  - Many modules are existing IP.
  - Some IP may be unmodifiable, other IP may be modified.
  - Some modules are created for the project.

# Characteristics of modern SoC designs

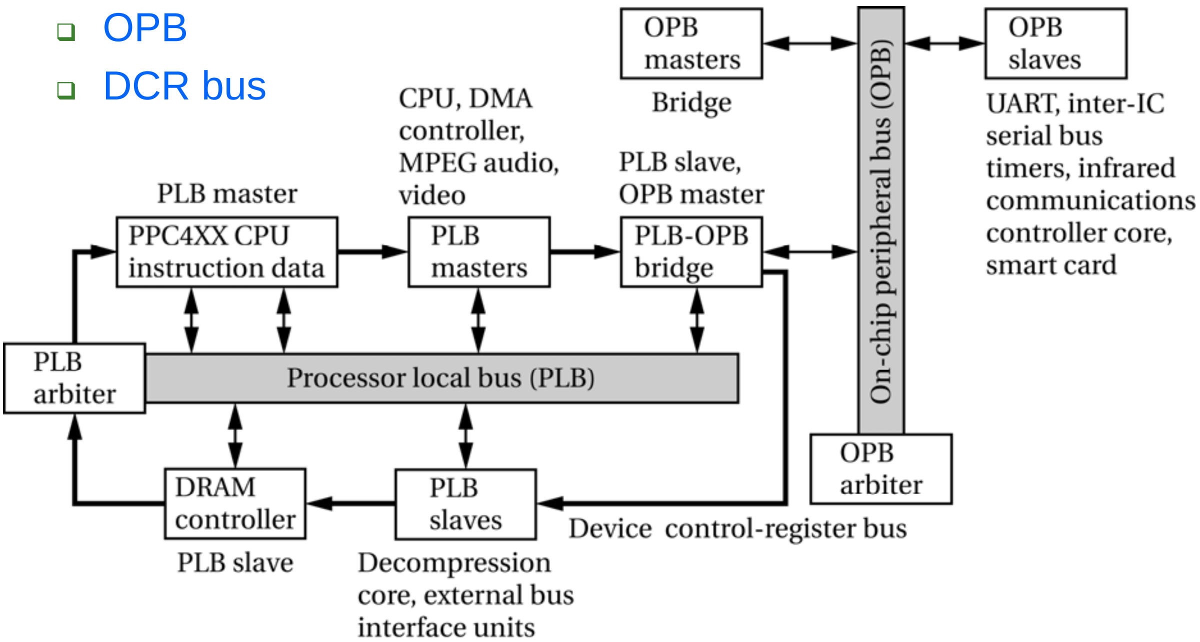
- Too big to be designed at register-transfer level.
  - CPUs running software.
  - Memory.
  - Devices.
- Too big to design all the IP blocks yourself.
- Too big to be verified solely by cycle-level simulation.

# Standard buses

- ARM AMBA
- IBM CoreConnect
- Sonics Silicon Backplaine
- VSIA (virtual socket interface alliance)
  - Defines a virtual component interface and a functional interface.
- Coware N2C and Cadence VCC
  - Provides tools for system-on-chip integration methodologies

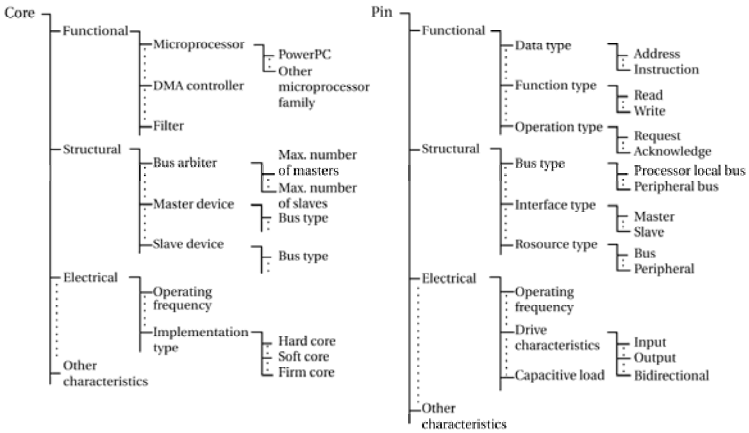
# IBM CoreConnect

- PLB
- OPB
- DCR bus

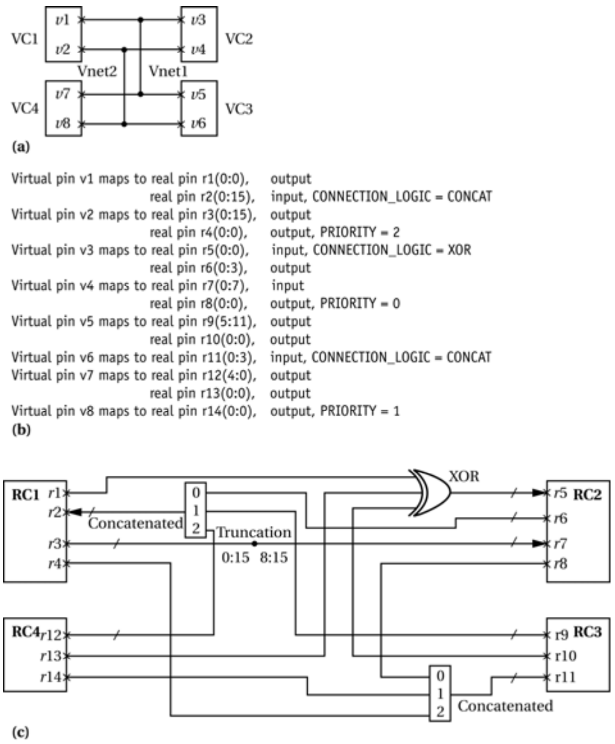

OPB masters
Bridge
OPB slaves
UART, inter-IC serial bus timers, infrared communications controller core, smart card
On-chip peripheral bus (OPB)
CPU, DMA controller, MPEG audio, video
PLB slave, OPB master
PLB master
PPC4XX CPU instruction data
PLB masters
PLB-OPB bridge
PLB arbiter
Processor local bus (PLB)
OPB arbiter
DRAM controller
PLB slaves
Device control-register bus
PLB slave
Decompression core, external bus interface units

# Coral (IBM tool) design methodology

- Virtual components describe a class of real components.
- Coral synthesizes glue logic between components.
- Interconnection engine generates netlist, checks designs.



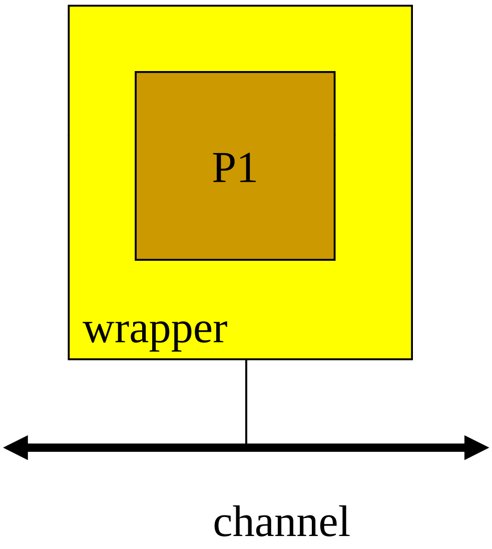
# Coral virtual-to-real synthesis



```
Virtual pin v1 maps to real pin r1(0:0),   output
                       real pin r2(0:15),  input, CONNECTION_LOGIC = CONCAT
Virtual pin v2 maps to real pin r3(0:15),  output
                       real pin r4(0:0),   output, PRIORITY = 2
Virtual pin v3 maps to real pin r5(0:0),   input, CONNECTION_LOGIC = XOR
                       real pin r6(0:3),   output
Virtual pin v4 maps to real pin r7(0:7),   input
                       real pin r8(0:0),   output, PRIORITY = 0
Virtual pin v5 maps to real pin r9(5:11),  output
                       real pin r10(0:0),  output
Virtual pin v6 maps to real pin r11(0:3),  input, CONNECTION_LOGIC = CONCAT
Virtual pin v7 maps to real pin r12(4:0),  output
                       real pin r13(0:0),  output
Virtual pin v8 maps to real pin r14(0:0),  output, PRIORITY = 1
```
(b)

# Component-based design

- Cesario/Jerraya: build MPSoCs from components to allow design reuse.
- Components + wrappers can be connected to channels.
- A wrapper is a design unit that interfaces a module to another module.
  - It can be HW or SW and may include both.
  - It can perform only low-level adaptation, such as protocol transformation



# Challenges in heterogeneous multiprocessors

- Multiple bus/network masters makes it harder to synchronize communications.
- Multiple busses/networks rather than single bus.
- Need specialized hardware for interprocess communication to offload the CPU.
- Need high-level communication primitives that can be off-loaded from CPU. (Shared memory I/O is too low-level)

# When a dedicated CPU is added,

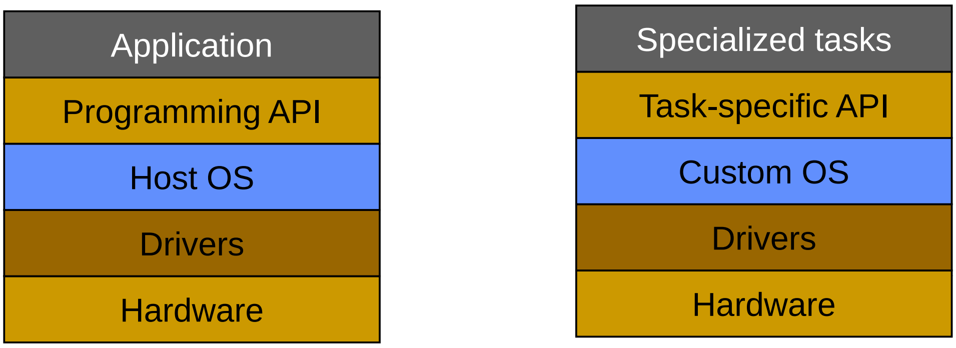
- Its software must be adapted in several ways.
  - The software must be updated to support the platform’s high-level communication primitives.
  - Optimized implementations of the host processor’s communication functions must be provided for interprocess communication
  - Synchronization functions must be provided.

# Challenges for EDA industry

- Must verify protocols, etc. without resorting to cycle-level simulation for everything.
- Chips will include several types of processors, making software development harder.
- Must adapt CPU, hardware IP blocks to the underlying communication fabric.

# Application vs. task-specific software stacks

- Host CPU and task-specific CPUs tend to run different stacks:



Application

Programming API

Host OS

Drivers

Hardware

Host CPU

Specialized tasks

Task-specific API

Custom OS

Drivers

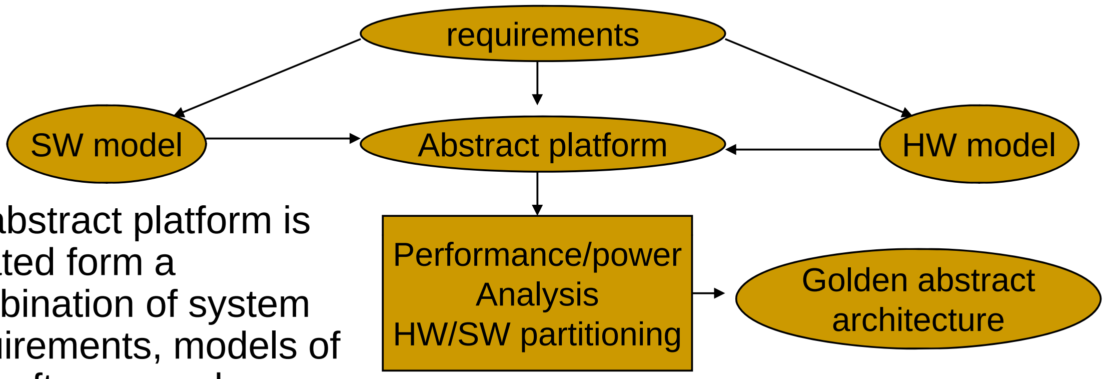
Hardware

Task-specific CPU

# Software adaptations for a dedicated CPU

- Adapt to hardware platform’s communication primitives.
- Provide optimized versions of host OS communication functions.
- Provide synchronization functions.

# System-level design flow (Jerraya et al.)



requirements

SW model

Abstract platform

HW model

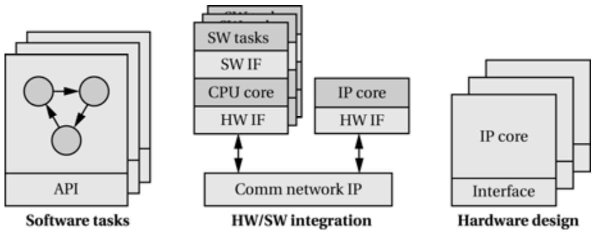
Performance/power
Analysis
HW/SW partitioning

Golden abstract architecture

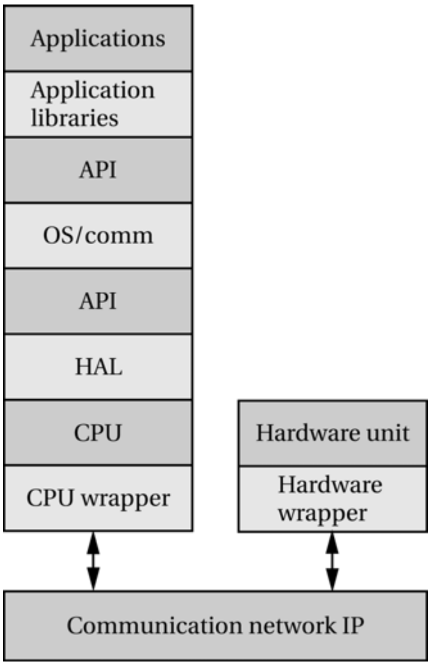
- An abstract platform is created form a combination of system requirements, models of the software, and models of the hardware components
- This abstract platform is analyzed to estimate the application's performance and power/energy consumption.
- Based on the results of this analysis, software is allocated and scheduled onto the platform.
- The result is a golden abstract architecture that can be used to build the implementation.

# Abstract architecture template

- Application libraries provide application-specific functions.
- OS and communication system provide scheduling and resource management.
- Hardware abstraction layer provides clock, interrupts, etc.
- CPU wrapper translates signals between CPU and network.



# Hardware and software abstraction layers



# Wrapper

- A wrapper-oriented design methodology presents several challenges
- Must be supported by tools that automatically generate the wrappers and deploy them within the architecture.
- Wrapper design is tedious and error prone when left to humans
- The wrapper generators must be able to build wrappers for several different types of protocols since realistic chips will use several types of interconnect.
- Some wrappers may have to interface two different protocols. the
- The wrapper generator must generate both SW and HW.
- Wrappers must be designed to support mixed-level co-simulation.

# Register-transfer implementation

- Given a golden architecture model, we still need to generate the complete register-transfer design of the hardware as well as the final software wrappers.
- The register-transfer design enumerates all the required components and connect them together.
- One of the major steps in register-transfer generation is the creation of the memory subsystem.
- This subsystem may include both connections between internal memory blocks and interfaces to the communication network.