Software Development of the EF2000 Flight Control Computers

Dr. H. Heusinger
EADS Deutschland GmbH
Military Air Systems

Agenda

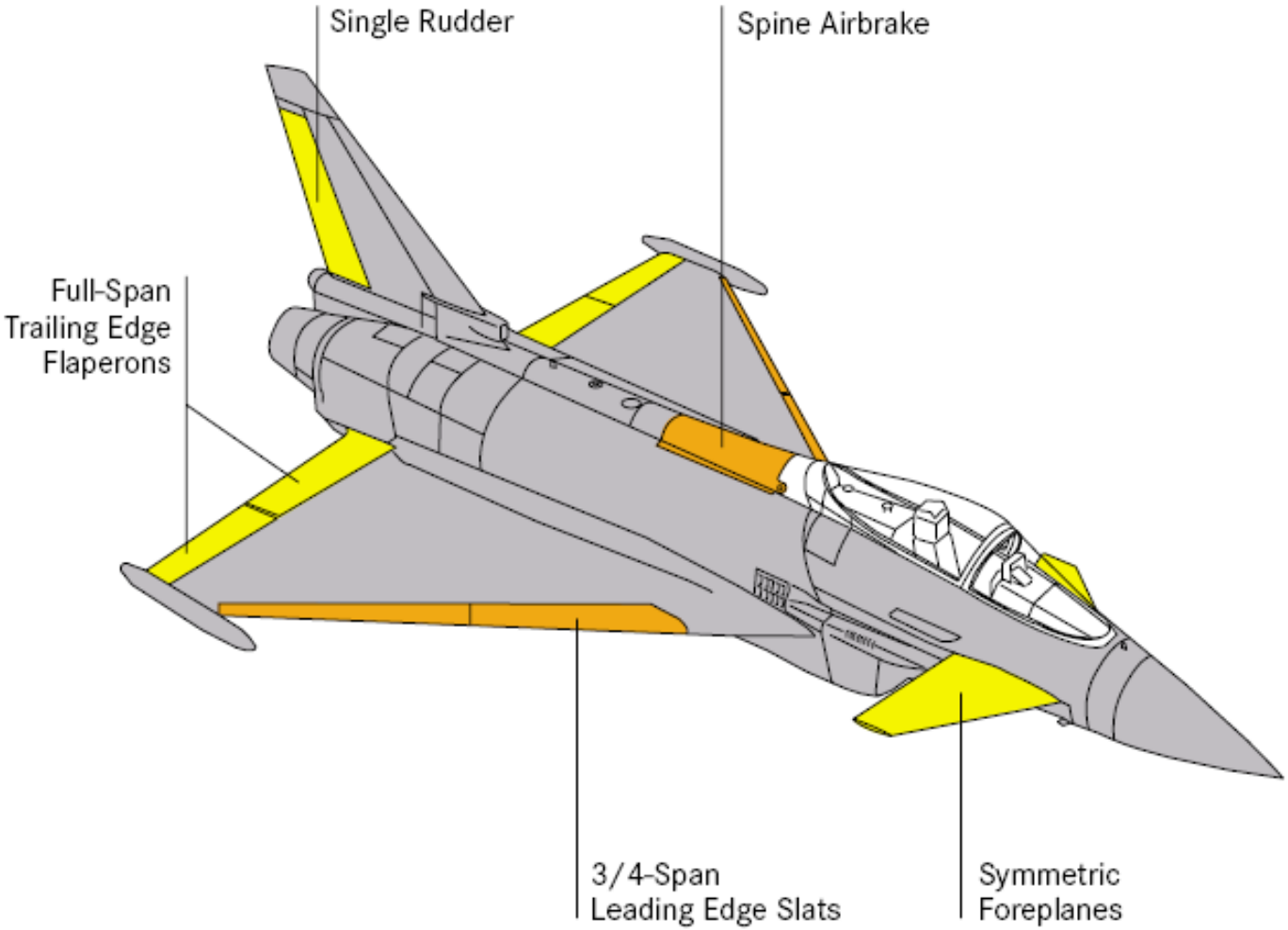
- Overview of Flight Control System
- Overview of Flight Control Computer
- Design Process
  - System Requirements
  - Software Requirements
  - Software Design
  - Test
- Tools

EF2000 Requirements

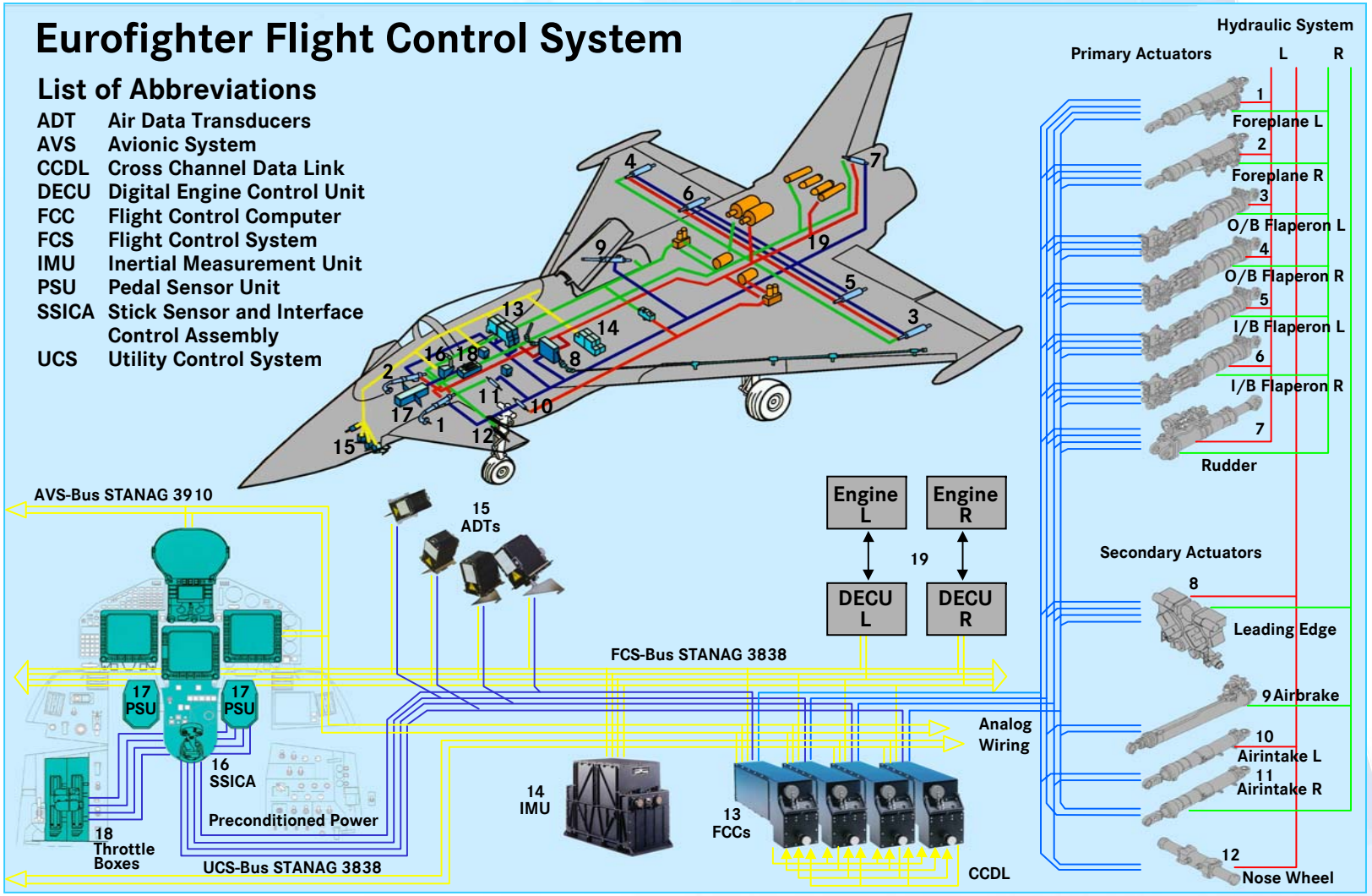
Goal: Achieve high survivability and mission reliability

- high agility
- carefree flying
- automatic recovery
- autopilot / autothrottle
- failure tolerant
- structural protection
- easy maintenance

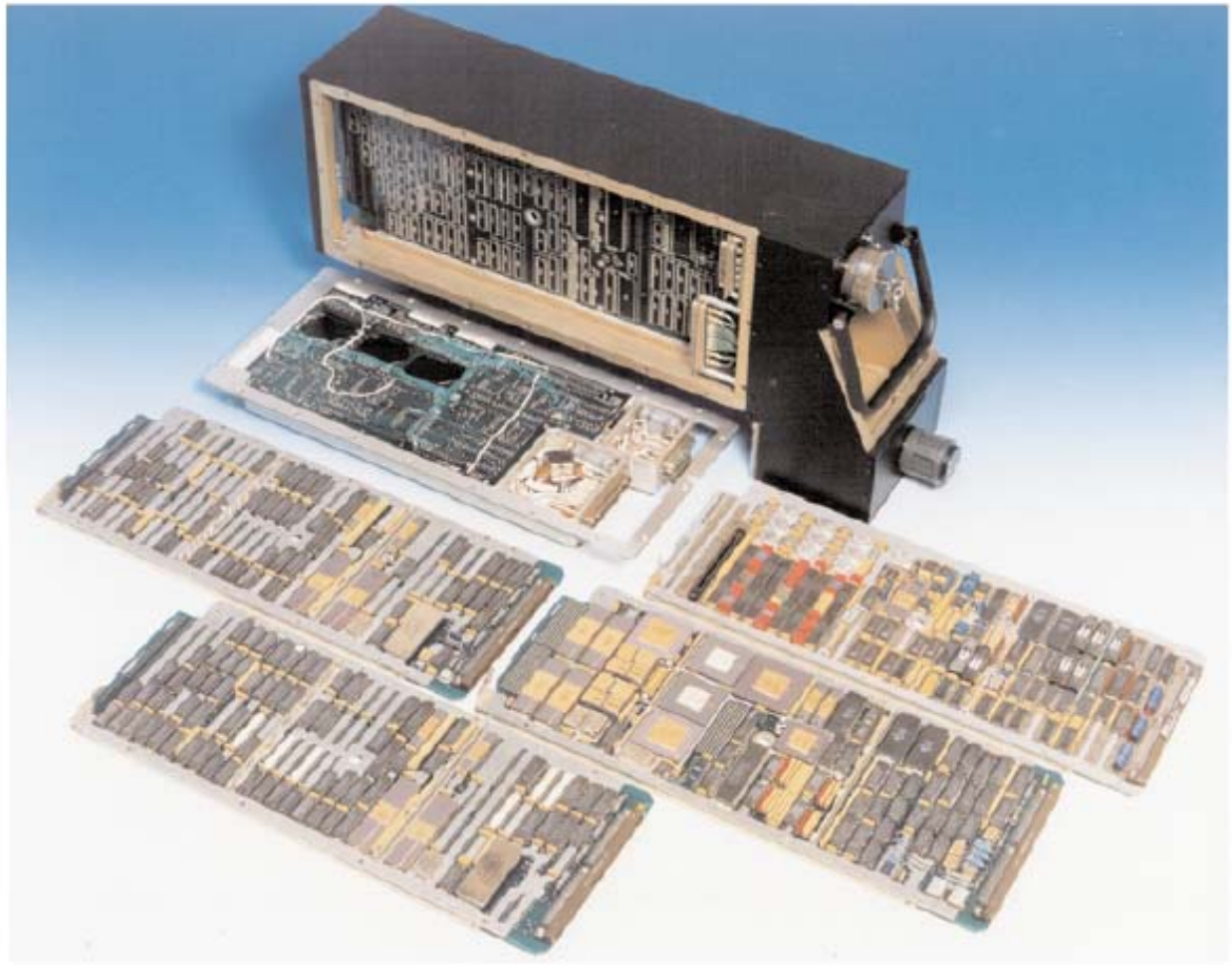
Aerodynamic Control Surfaces



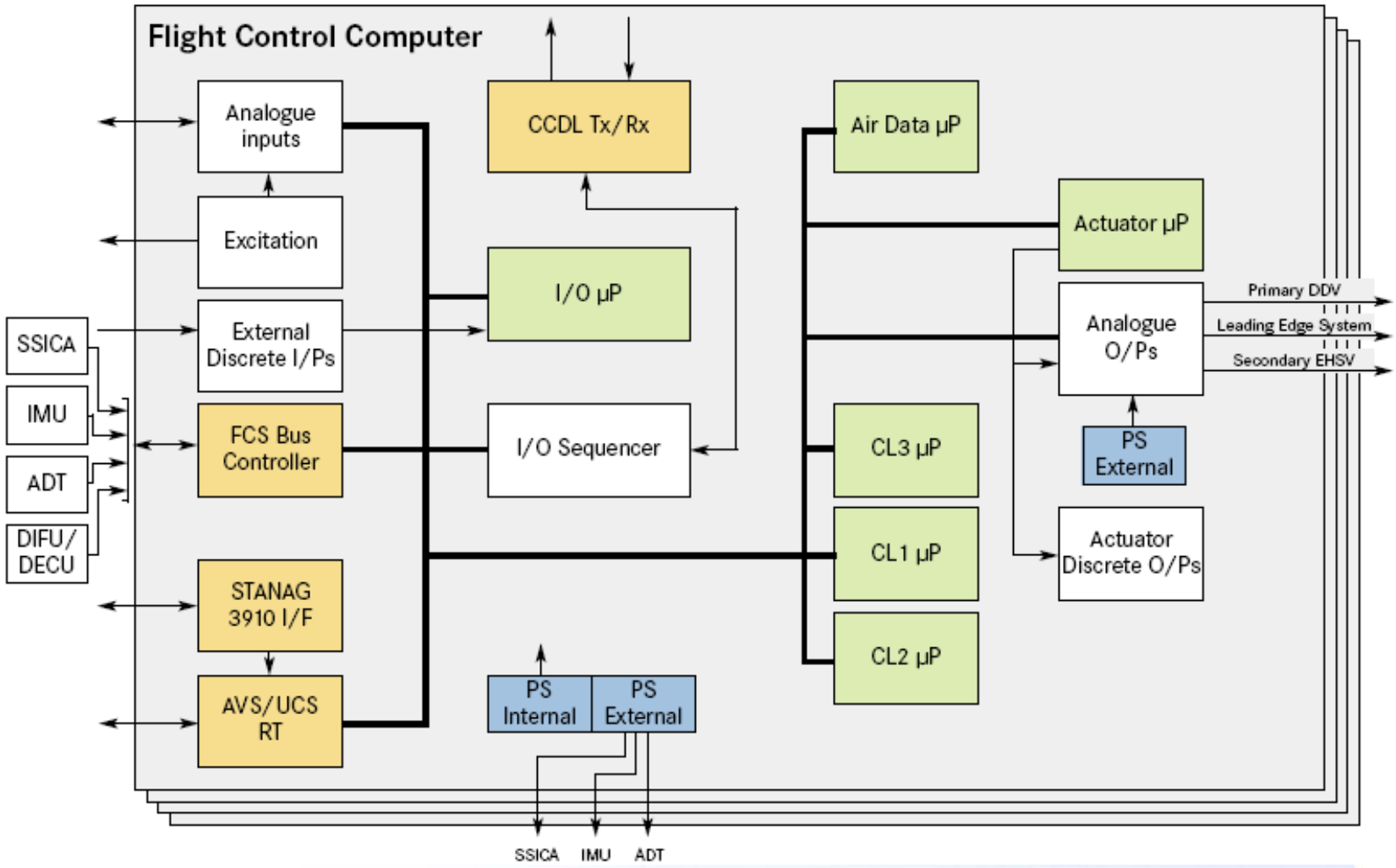
EF2000 Flight Control System



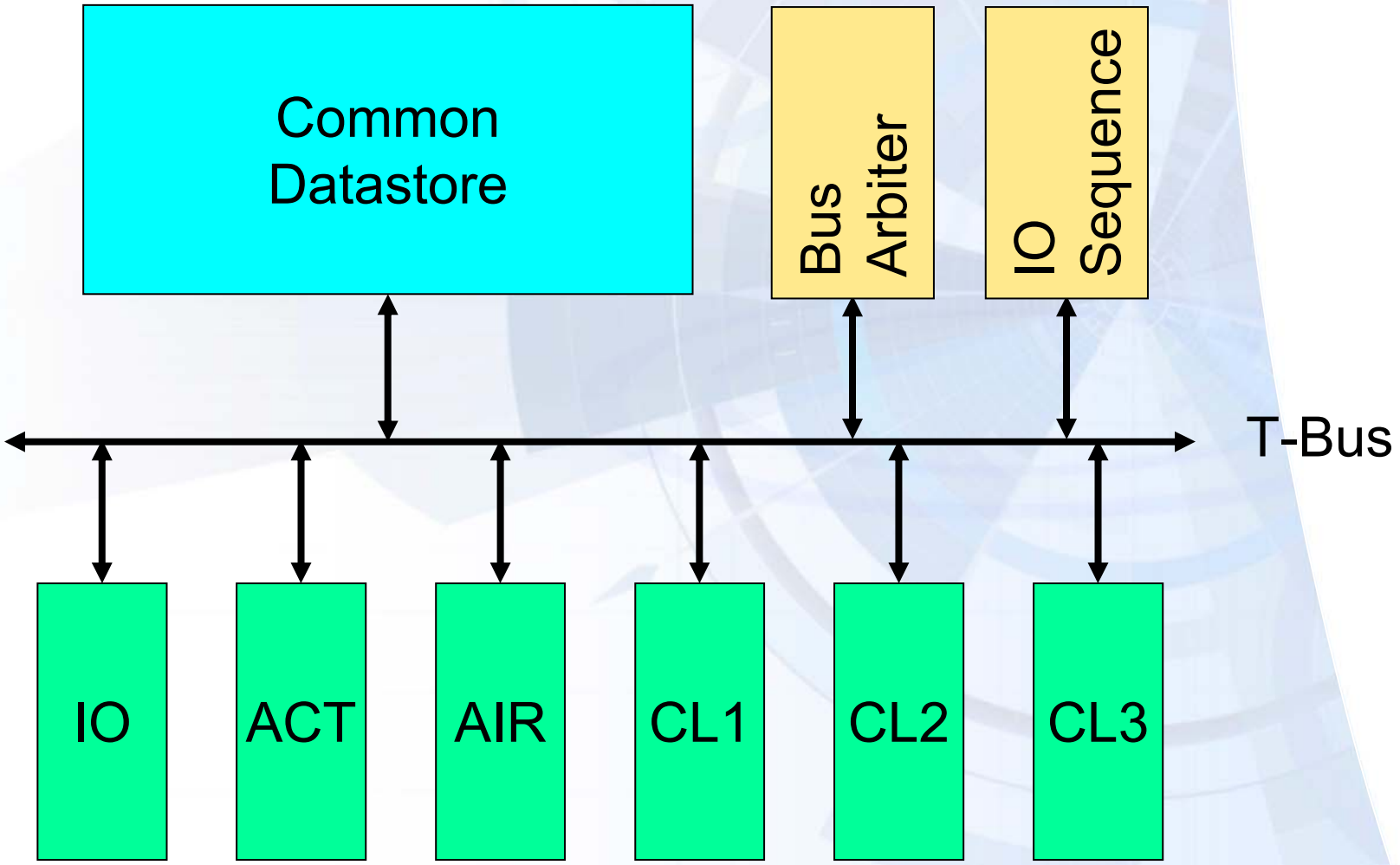
Flight Control Computer



Flight Control Computer Architecture



Inter Processor Communication



General FCS Requirements

- A single failure must not endanger the mission success
- A second failure must not generate a safety hazard
- All FCCs shall be interchangeable
- All FCCs shall generate the same actuator demands at all times (even with failures)
- There must be an instantaneous reaction to the pilot's inputs

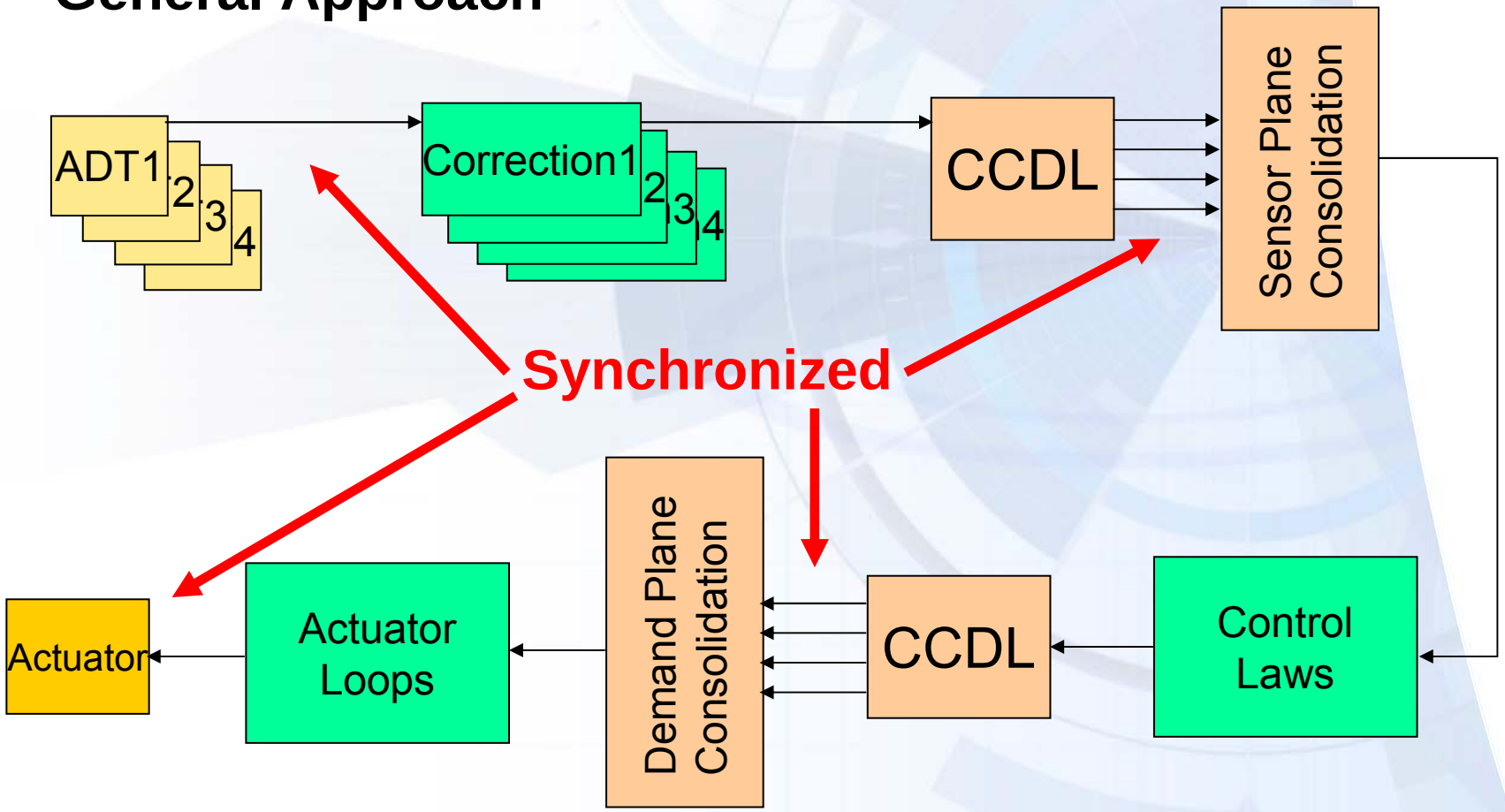
Interchangeable FCCs

Into which slot fits this FCC?

- FCCs must be interchangeable
- FCCs must have identical software
- FCS must verify the HW / SW standard
- FCC must take over stored data from others

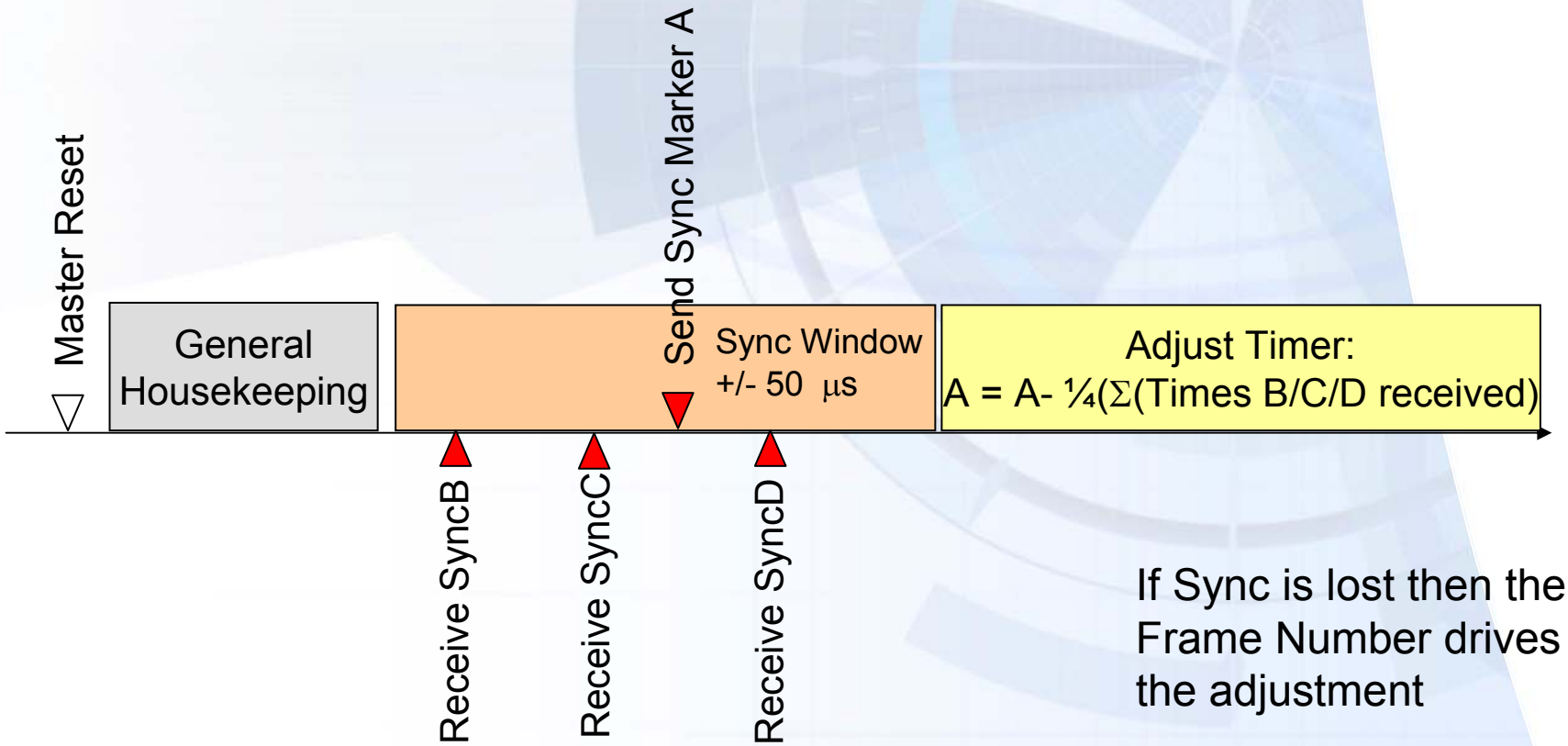
Generate Identical Actuator Demands

General Approach

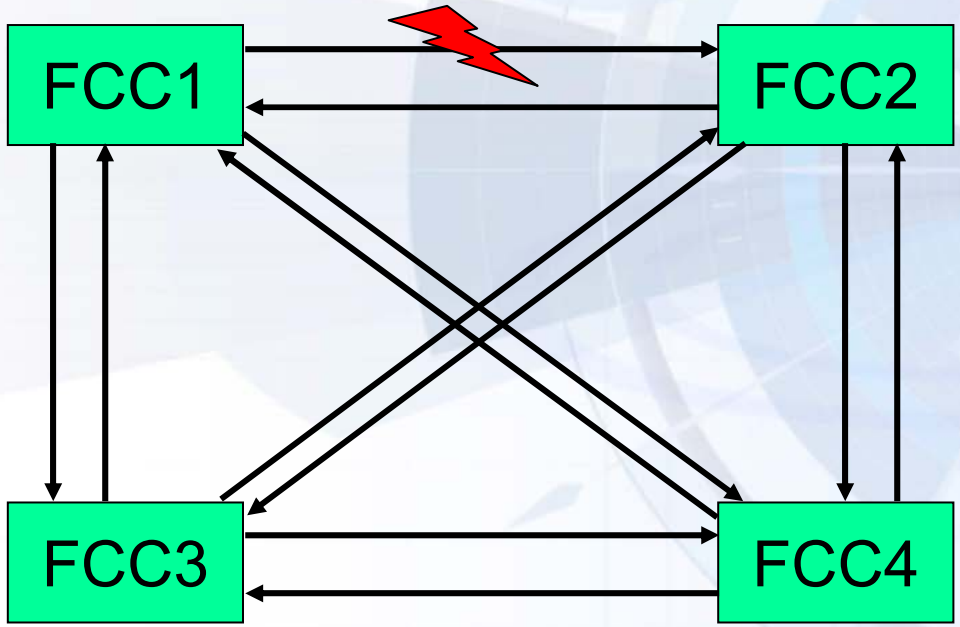


Synchronisation

- Each FCC generates a master interrupt every 12.5 ms.
- The FCC tries to keep the timer interrupts within 100 μs.



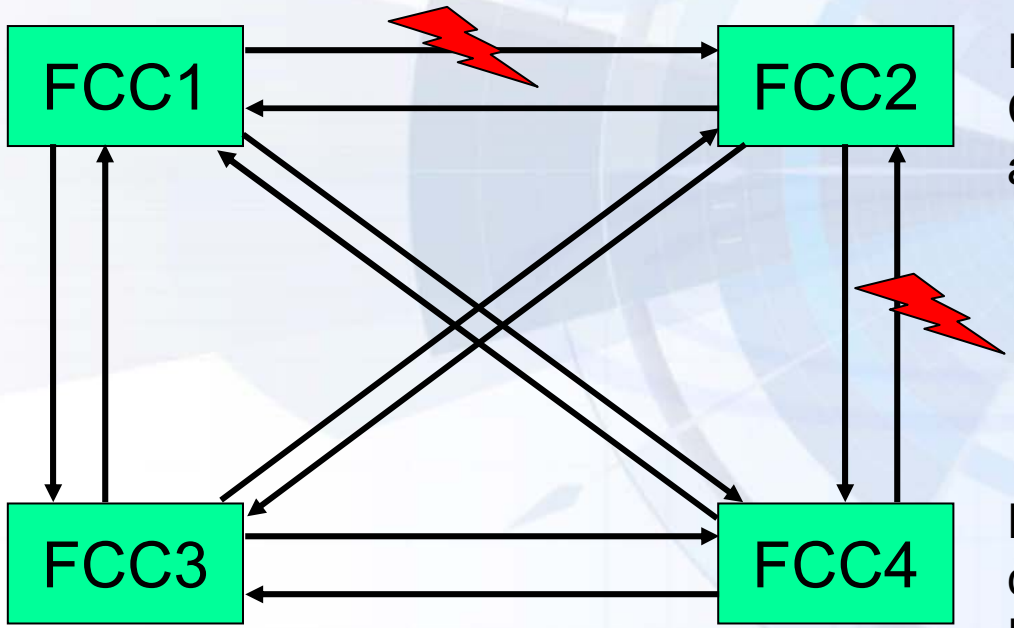
Channel Exclusion (part 1)



FCC2 detects CCDL fail from FCC1 due to missing checkword

FCC2 tells FCC4 that it cannot see FCC1 but all others; FCC4 assumes FCC1 transmitter faulty

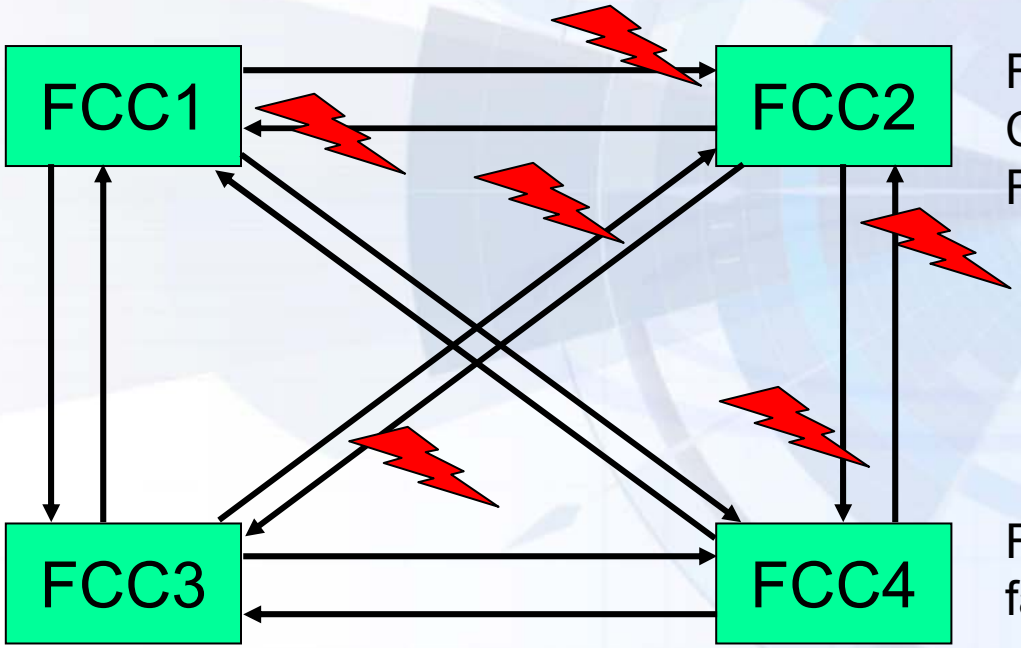
Channel Exclusion (part 2)



FCC2 detects CCDL fail from FCC1 and FCC4

FCC2 tells FCC4 that it can't see FCC1 and FCC2; FCC4 assumes FCC2 receiver faulty and works quadruplex

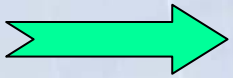
Channel Exclusion (part 3)



FCC2 detects
CCDL fail from all others
FCC2 isolates itself

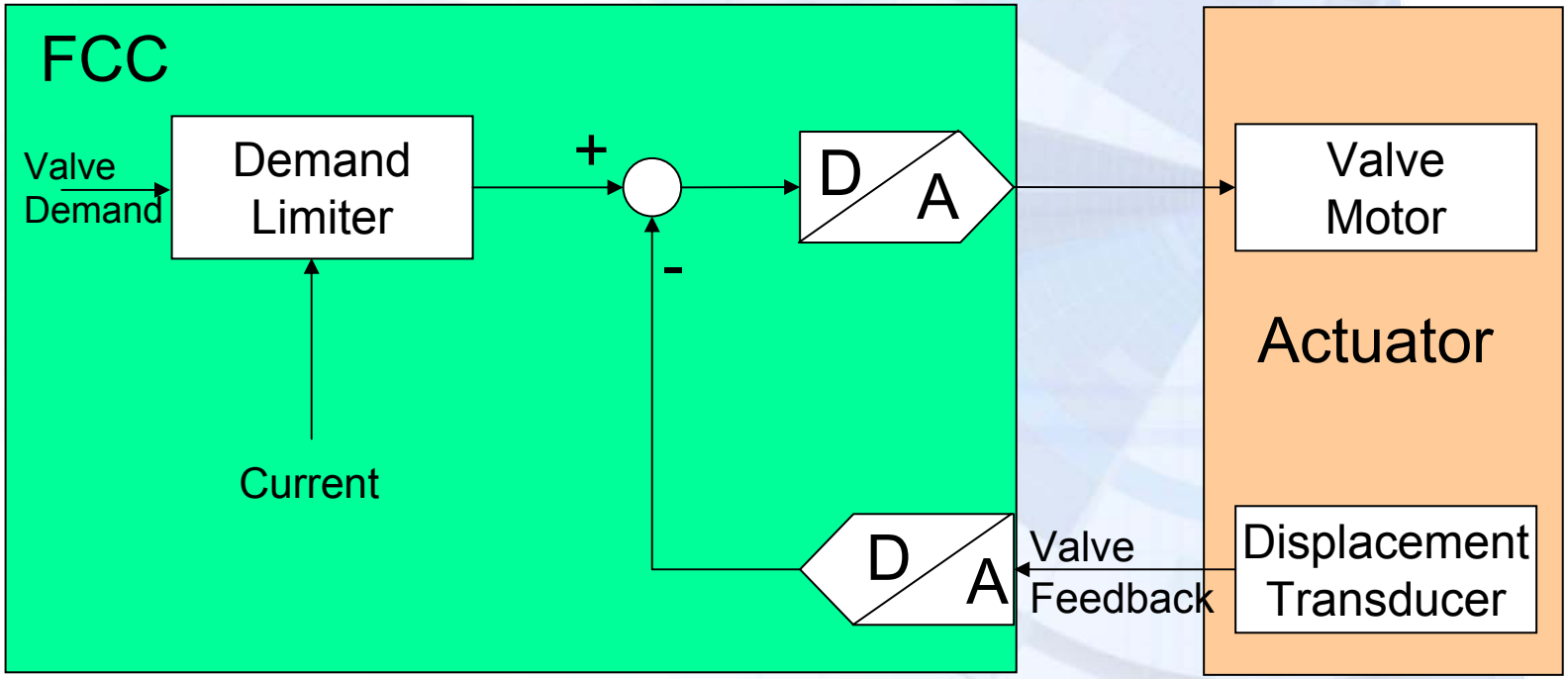
FCC4 detects CCDL
fail from FCC2

Software Design Process

SW Requirements Analysis		FCC Algorithms Timing Requirements
SW Top Level Design		Ada Package Descriptions: • Allocation to Processors • Scheduling of Tasks
SW Detailed Design / Code		Ada / Assembler Code • Implementation of functions • Optimisations

Software Requirement (Example 1)

The actuator inner loop shall implement the following:

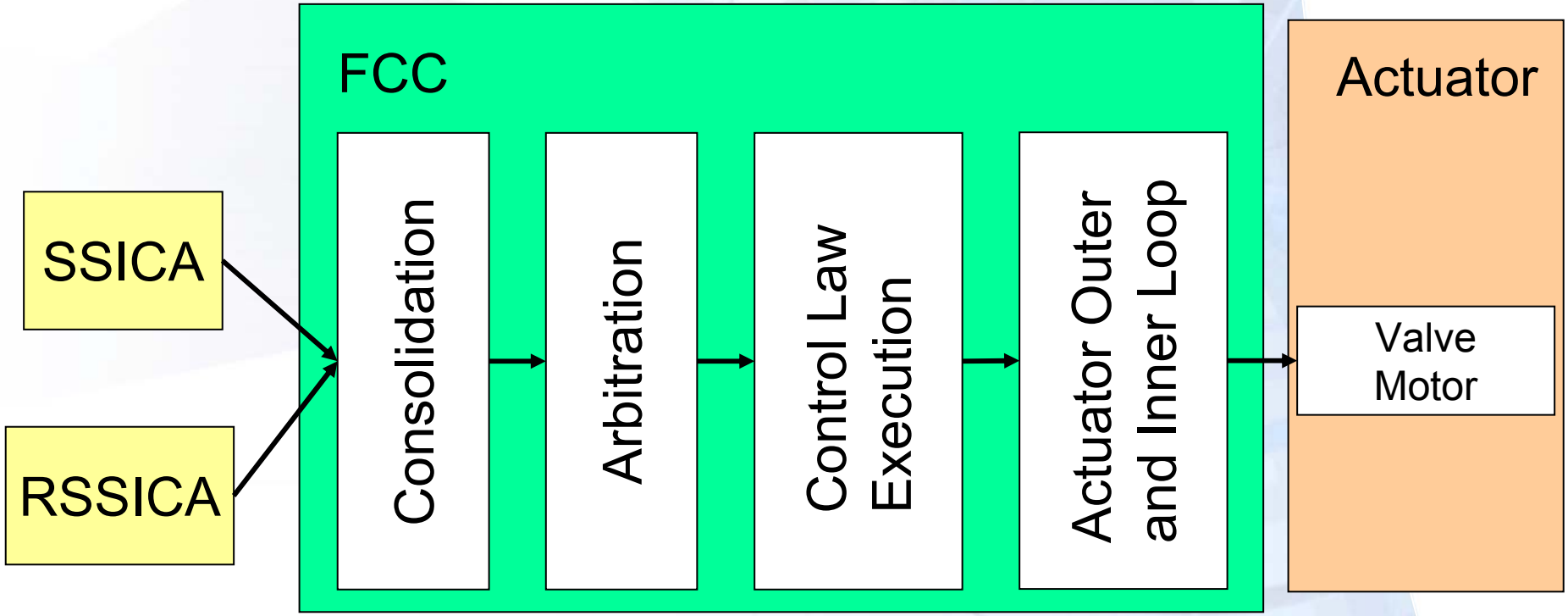


Iteration Rate: 320 Hz

Delay from Valve Feedback to Motor Demand ≤ t_1 ms

Software Requirement (Example 2)

The pilot's inceptor demands shall flow through the system as:



Iteration Rates: 40 Hz; 160 Hz; 320 Hz

Fastest Delay from SSICA/RSSICA to Motor Demand ≤ t_2 ms

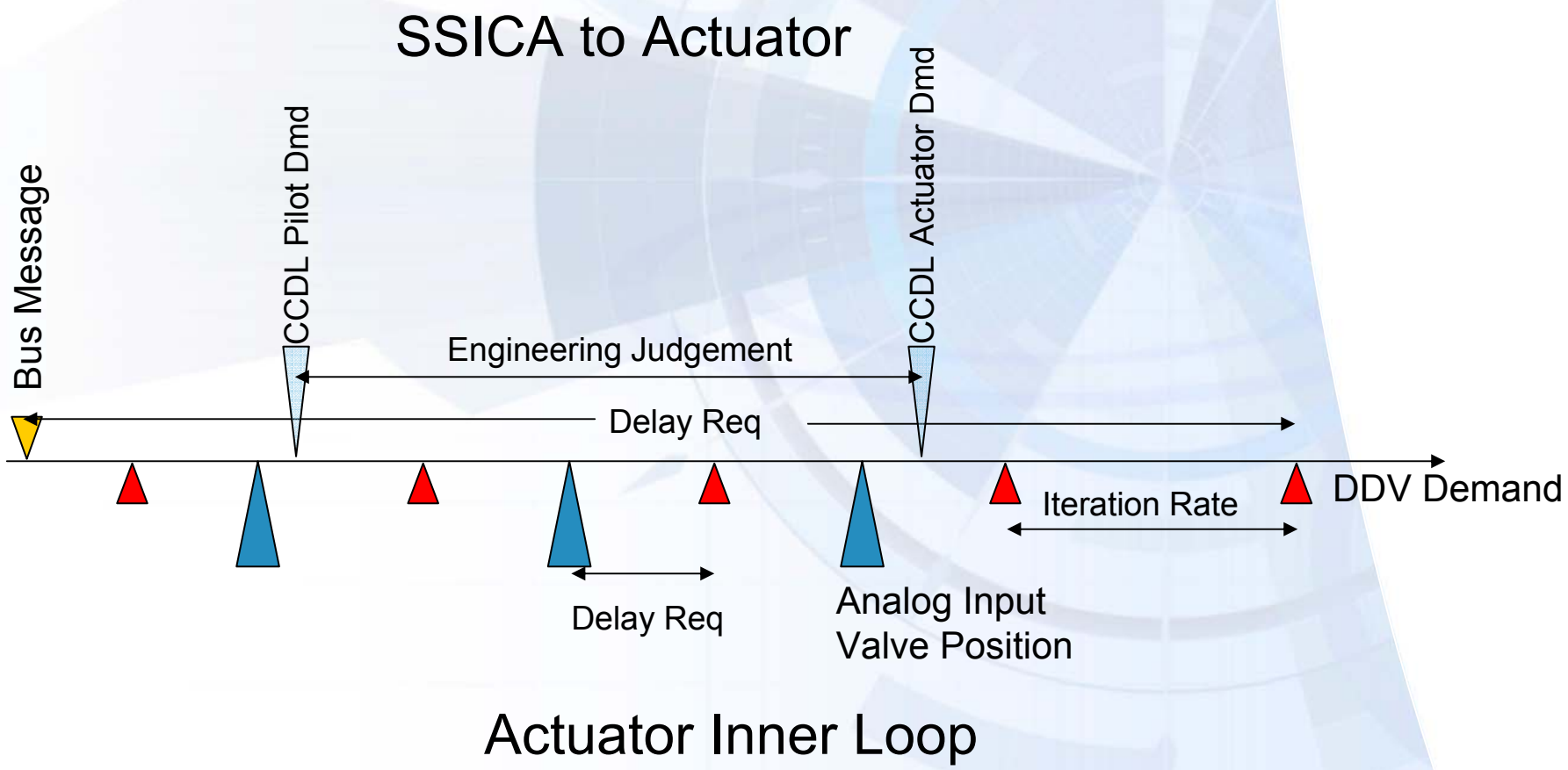
Software Top Level Design

Tasks:

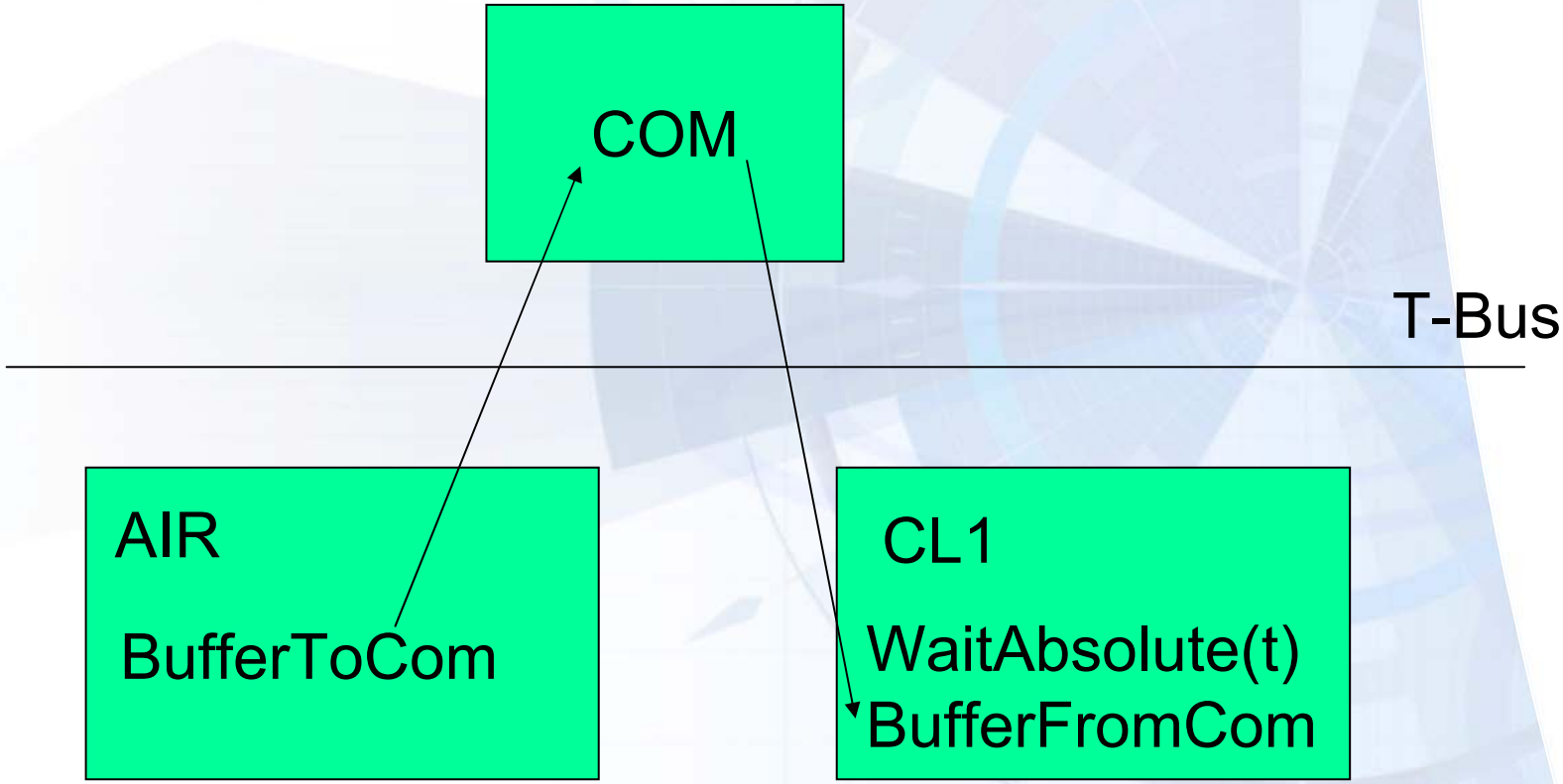
- Define IO Sequence
- Allocate the functions to the processors
- Define the Ada Package Specifications
- Define the communication between the processors
- Estimate the worst case run time, the program store and the needed RAM for each module
- Generate the schedulers for the modules

Software Top Level Design (IO Sequence)

Principle: Just in time is in time



Software Top Level Design (Communication)



Worst Case Time when BufferToCom is finished < t

Software Top Level Design (Scheduling)

Functional Chain

with Delay Requirement

640 Hz

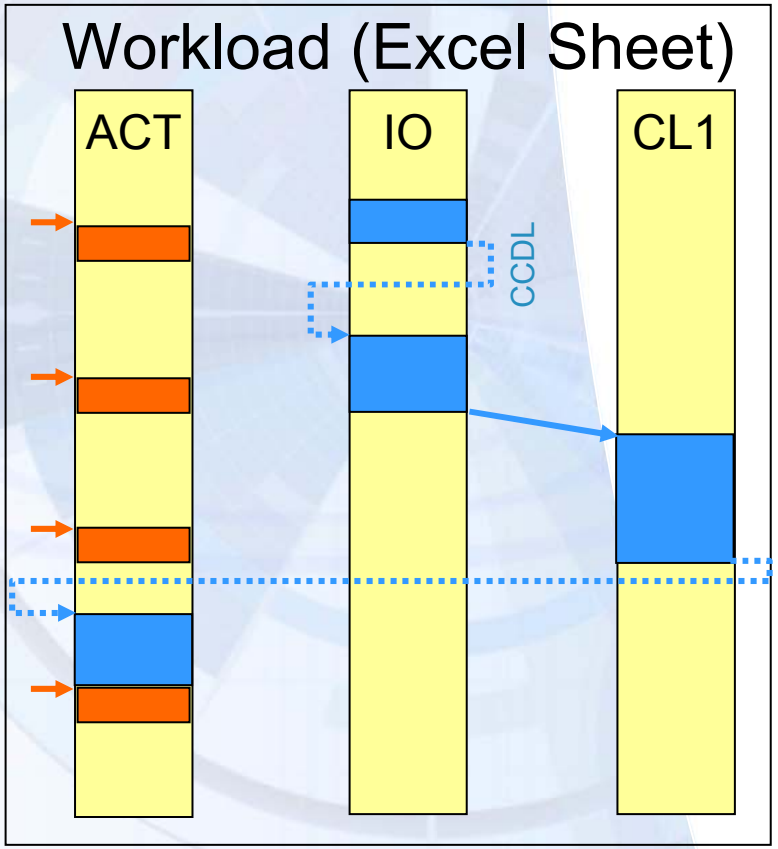
Iteration Rate

1.25 Hz

Act Inner Loop

SSICA to Act

without Delay Requirement



Define the buffer modules now

Software Coding (Standard)

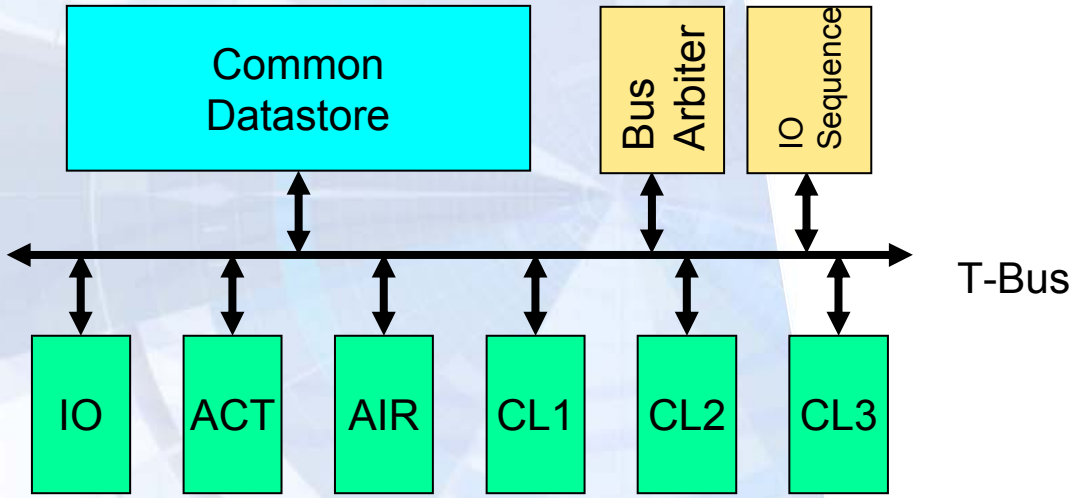
- SafeAda must be used
  - No tasking
  - No predefined exceptions
  - No pointers
  - No unchecked conversion
  - No recursive calls
  - etc.
- Assembler should not be used
  - runtime system
  - highly used components (e.g. voter / monitors, filters, interpolations, etc.)
  - allocation of memory

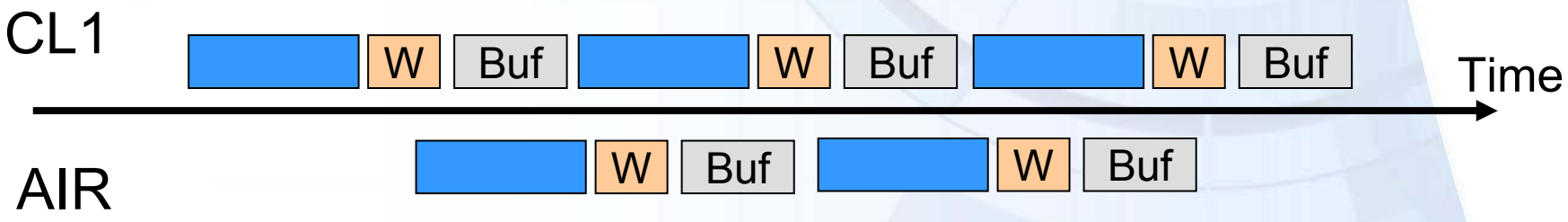
Software Coding (Optimisations)

Waits can speed up the program!!

Access to COM slow:

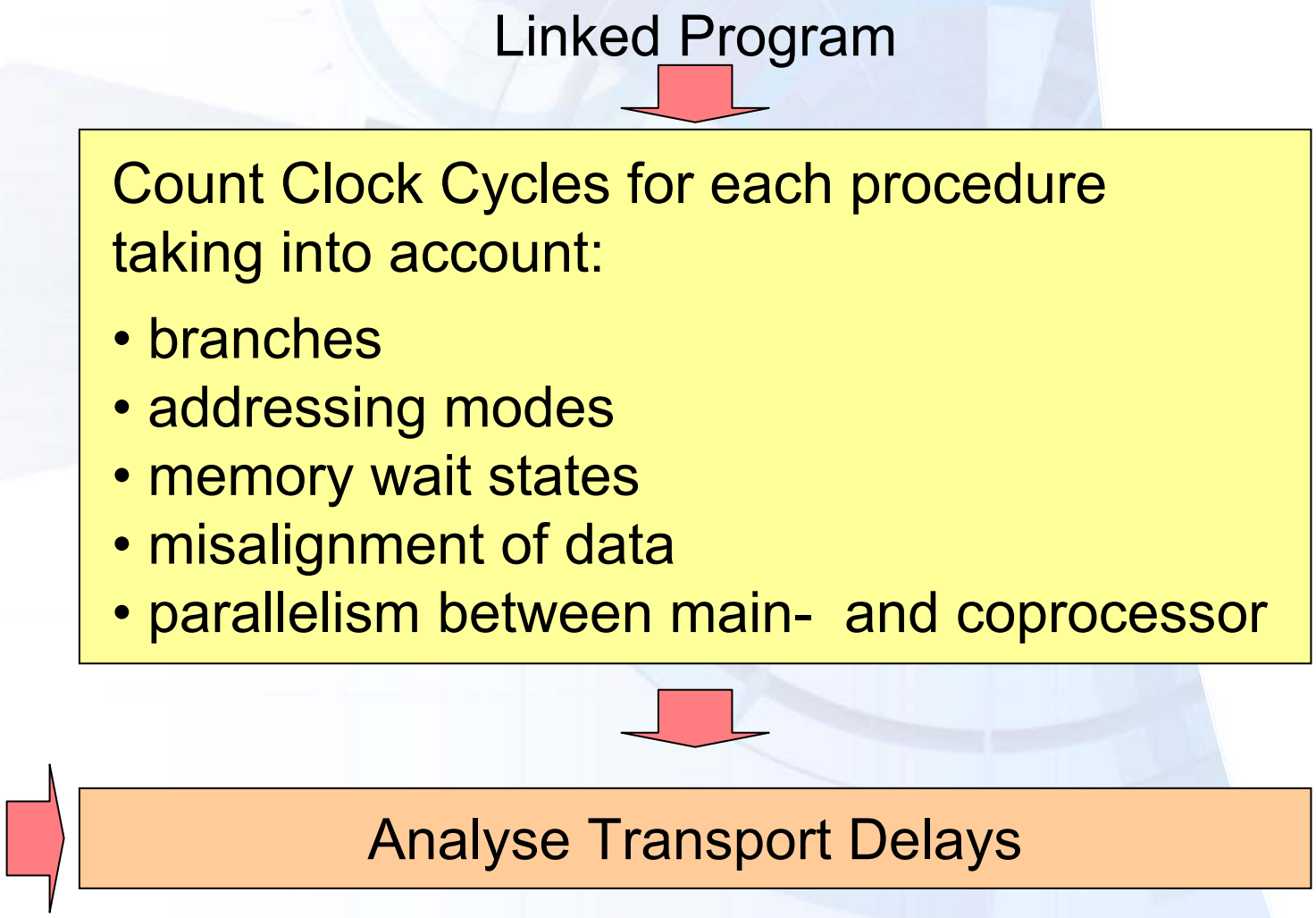
- slow T-Bus
- 6 bus clashes





➡ Waits guarantee that at most 3 bus clashes occur

Software Coding (Run Time Analysis)



Software Coding (Hazard Analysis)

Prove that no hazard is introduced due to the coding

- Divide by zero Analysis
- Overflow Analysis
- Denormalisation Analysis
- Memory Usage Analysis
- Ada / Assembler Interface Analysis
- Iteration Rate Analysis
- Illegal Instruction Analysis
- Recursion Analysis
- Compiler Defect Analysis
- etc.

Software Coding (Limited Change Capability)

Difference of Object Code drives regression test

small change of source → small change of linked program

We must be able to fix code and data in memory

Sometimes it's better to move unchanged data rather than changed data

Changes of the runtime and data flow must be handled locally

Software Testing

Unit Tests

Verify that each procedure implements the design given in Top Level / Detailed Design
(Statement, Branch and MC/DC coverage on target)

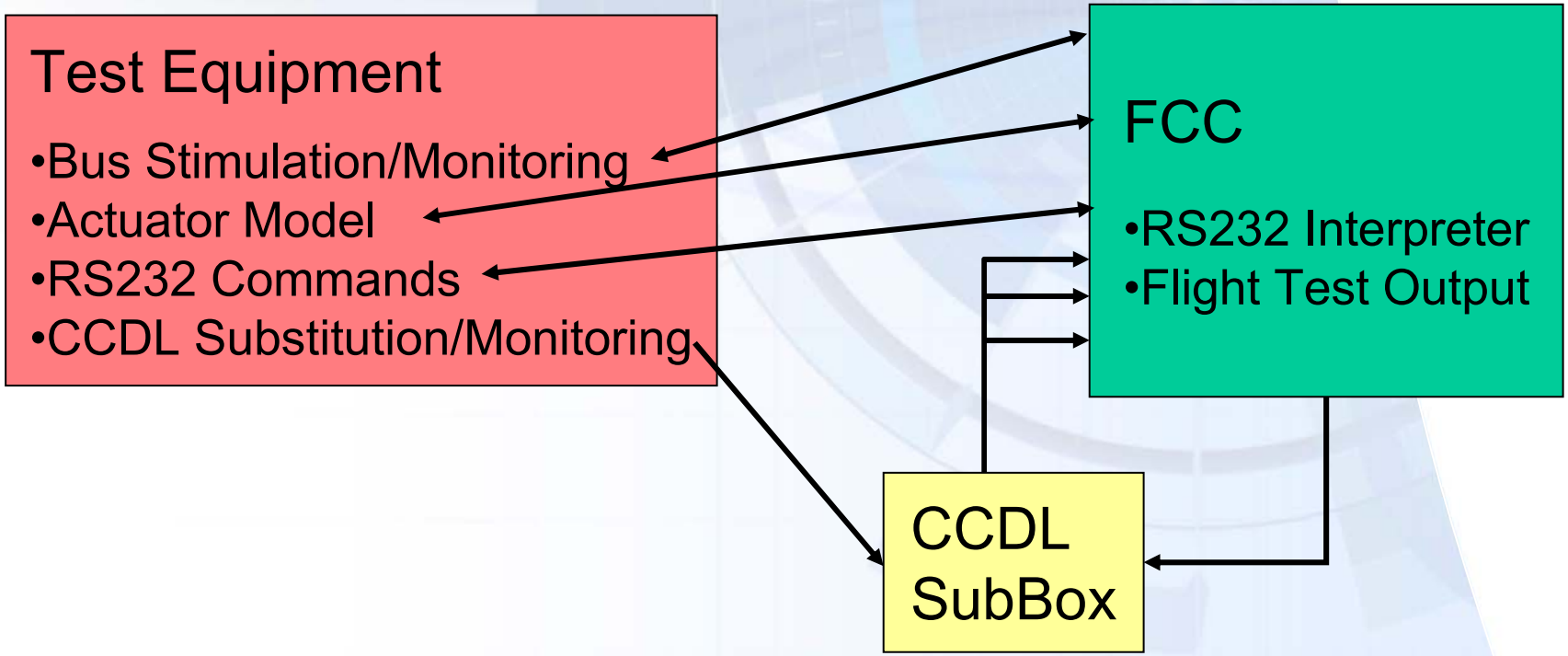
Software Integration Tests

Verify that all calls to procedures are correct according to the Top Level / Detailed Design

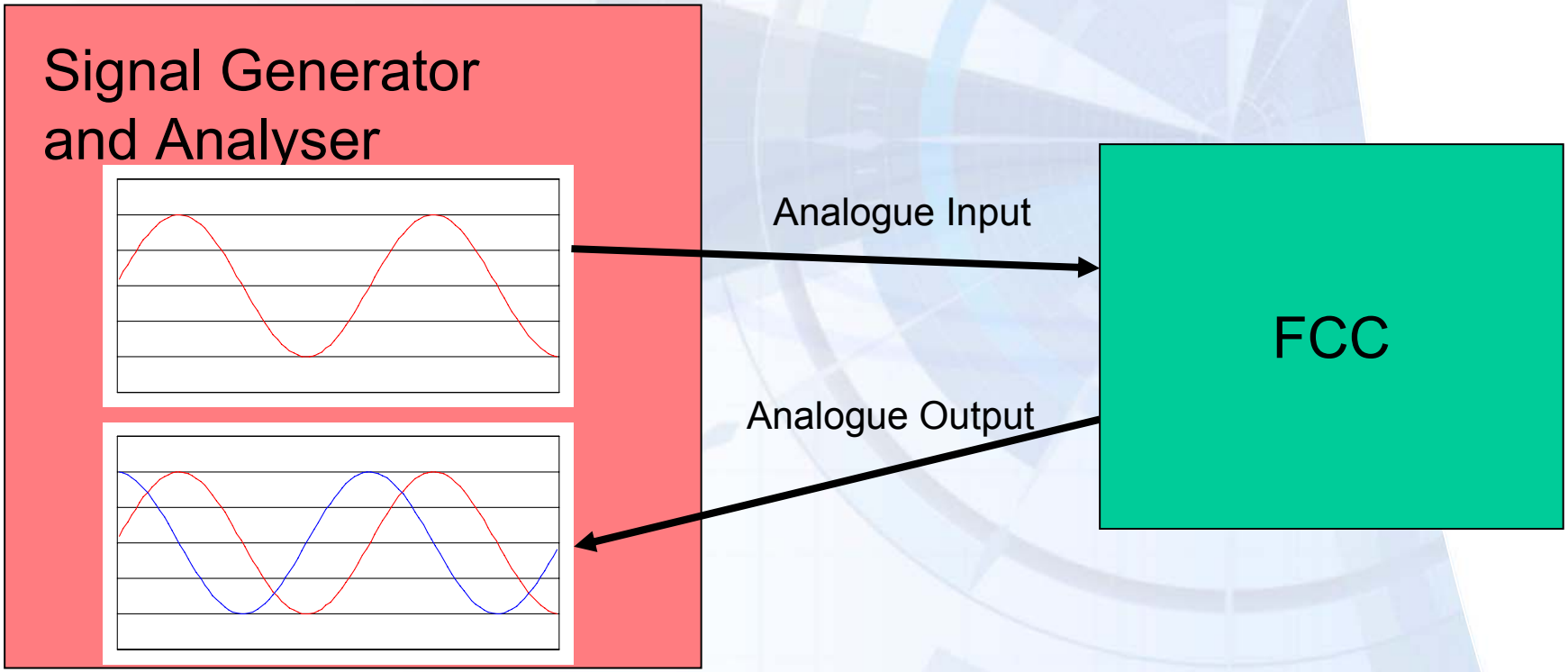
Software Testing

Hardware / Software Integration Tests

Verify that each software requirement is correctly implemented in the actual

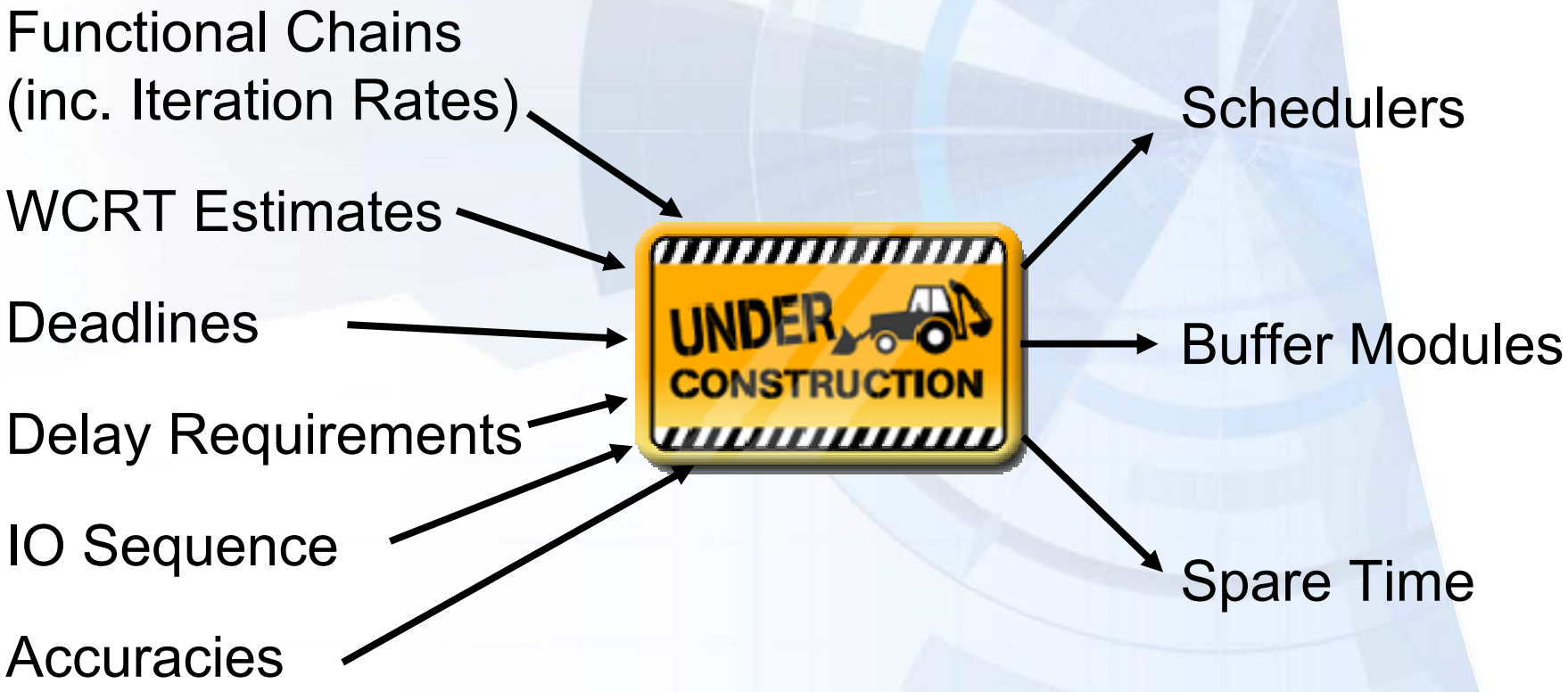


Software Testing (Timing)



Tool Support

Generator for Schedulers



Tool Support (Validated)

Worst Case Runtime Analyser

- new processor MPC 565
- cache enabled
- runtimes with parameters

Transport Delay Analyser

- graphical interface to data flow diagrams
- automatic input of IO sequence
- automatic bus clash analysis

EADS DEFENCE & SECURITY

Q & A