Automatic implementation of TTEthernet-based time-triggered avionics applications

Raul A. Gorcitz, Thomas Carle, David Lesens, David Monchaux, Dumitru Potop, Yves Sorel

(CNES, INRIA, Airbus DS)

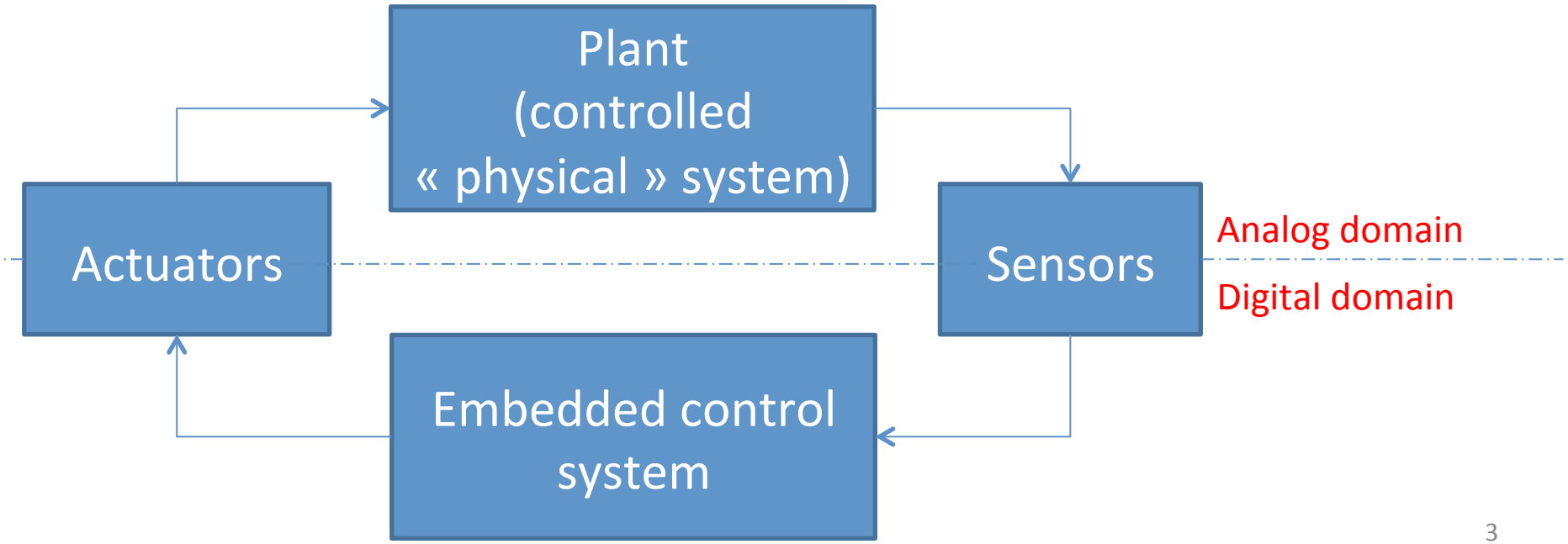
dumitru.potop@inria.fr

Open question from Nicolas

- The end of time in time-triggered systems research ?
- Maybe not: Time-triggered systems and static scheduling are twin fields
  - Time-triggered as an abstraction (which it is anyway to some point, except in fully synchronous HW)
  - Platform abstraction issues
    - "Simplifying hypotheses" must go away at some point, especially the unsafe ones
      - » No synchronization time, no interferences due to memory access, etc.
    - Precision
  - Scalability issues:
    - Will constraint solvers scale on 15000 elementary bocks, on 16 cores, taking into acount allocation/scheduling/memory interferences/etc. ?
  - My take: RT scheduling seen as compilation

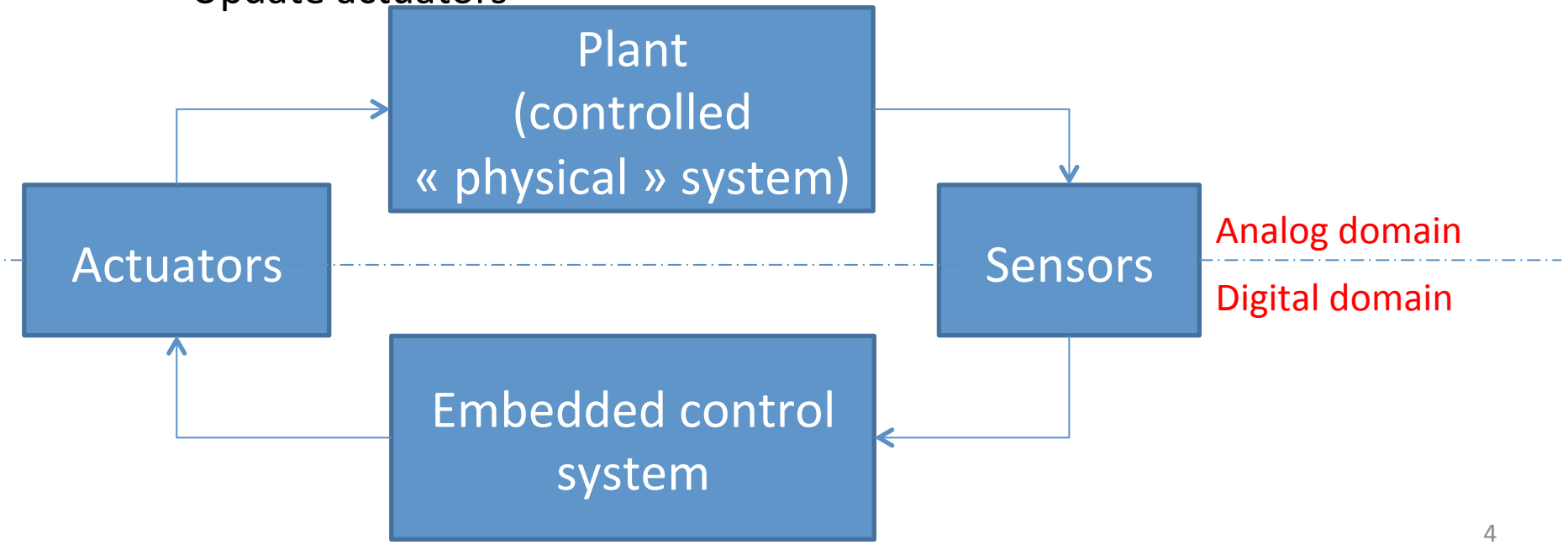
Embedded control systems

- Computing system that controls the execution of a piece of « physical » equipment, in order to ensure its correct functioning

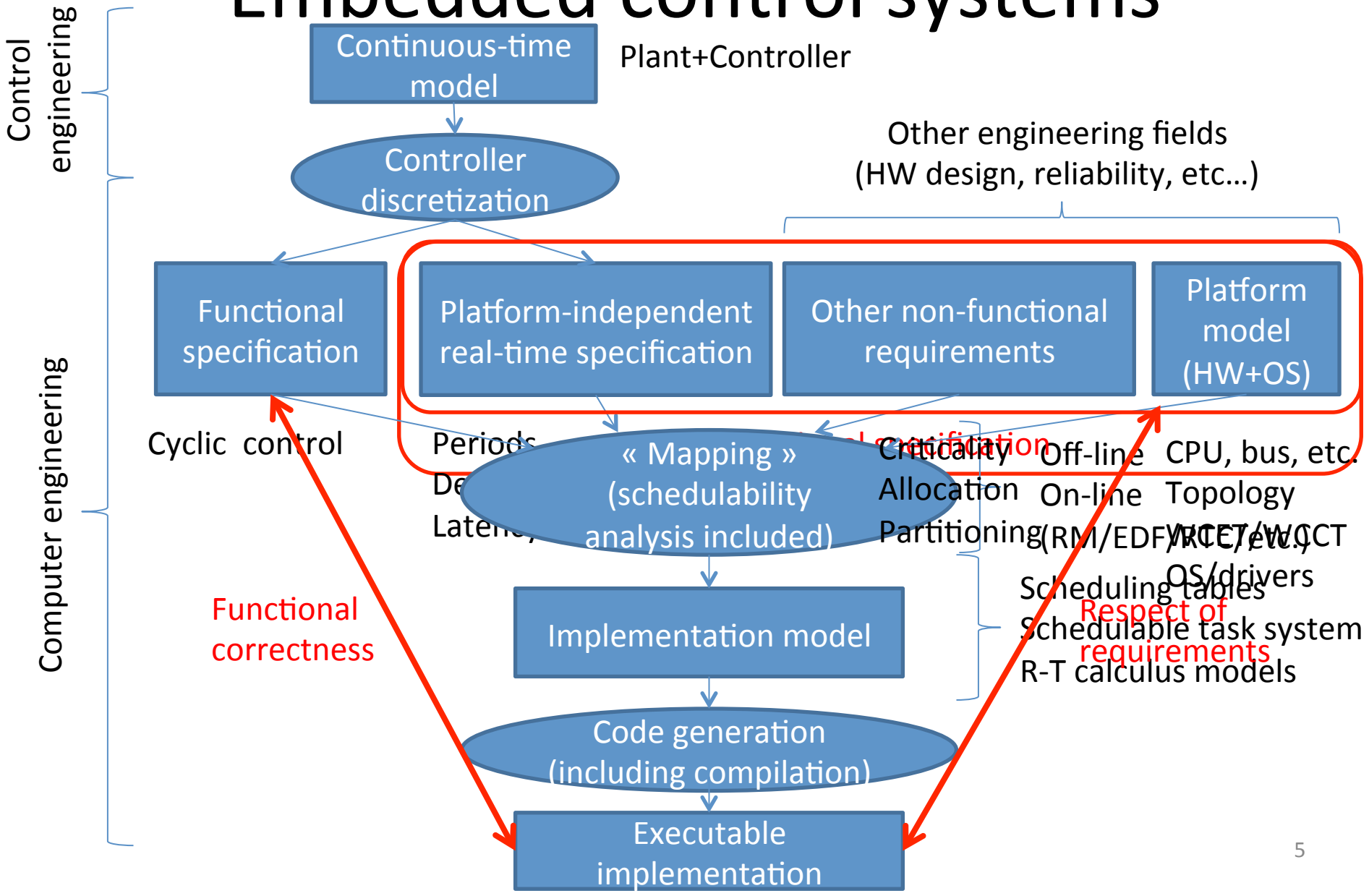


Embedded control systems

- Example: Guidance, Navigation and Controle (GNC) in a plane/ launcher.
  - Use sensors (GPS, accelerometers, gyros, etc…) to determine the position, speed, and attitute of the plane/rocket
  - Compute the correction (control algorithm)
  - Update actuators



Embedded control systems



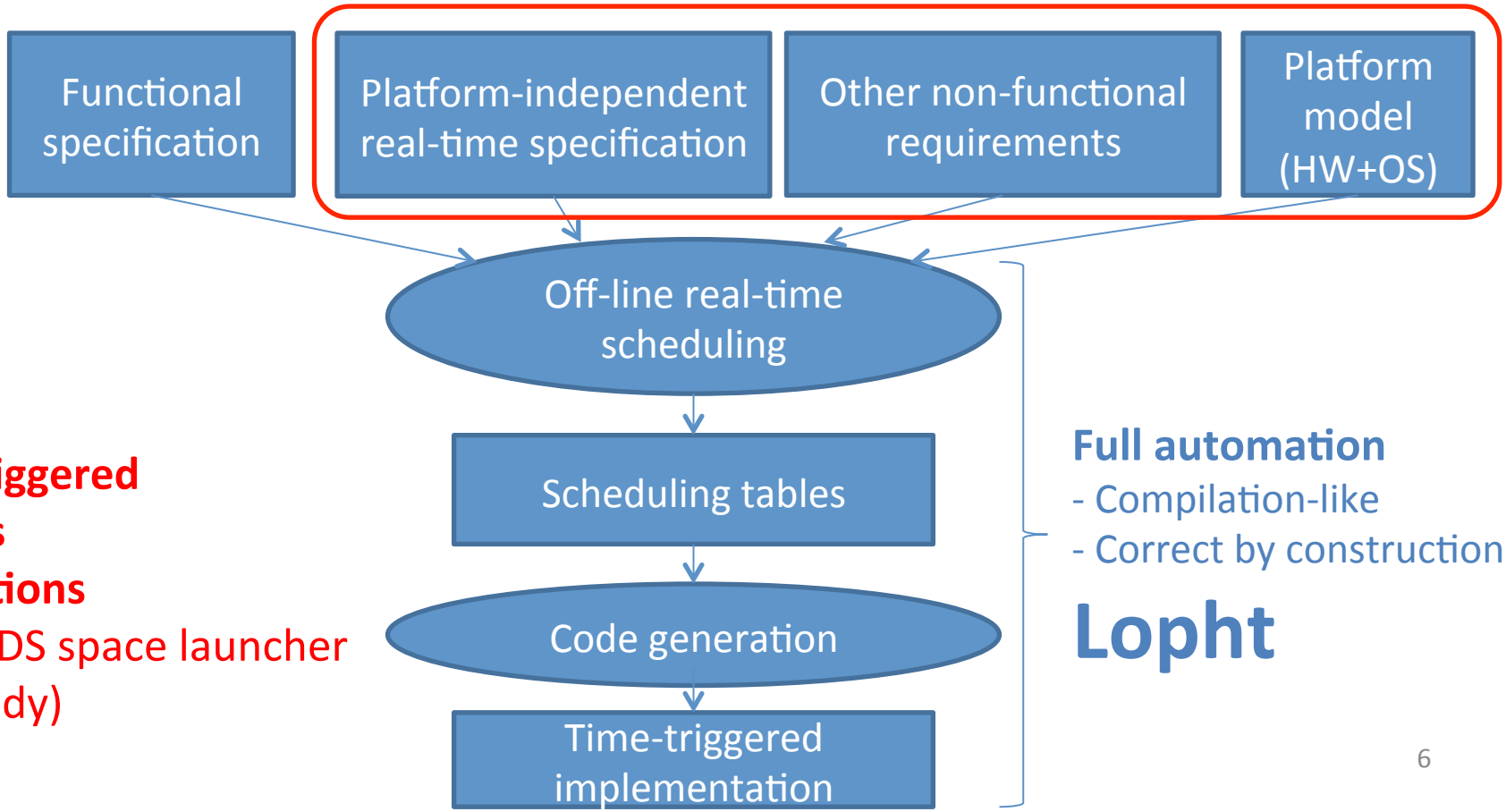
Our objective

Synchronous specifications (Scade/Heptagon)

Release dates
Deadlines
(periods, latency constraints)

Allocation
Partitioning
(criticality)
Preemptability

Topology (CPUs,buses,…)
WCET/WCCT
IMA/ARINC 653 (VxWorks 653)
TTEthernet



Time-triggered avionics applications
(Airbus DS space launcher case study)

Full automation
- Compilation-like
- Correct by construction

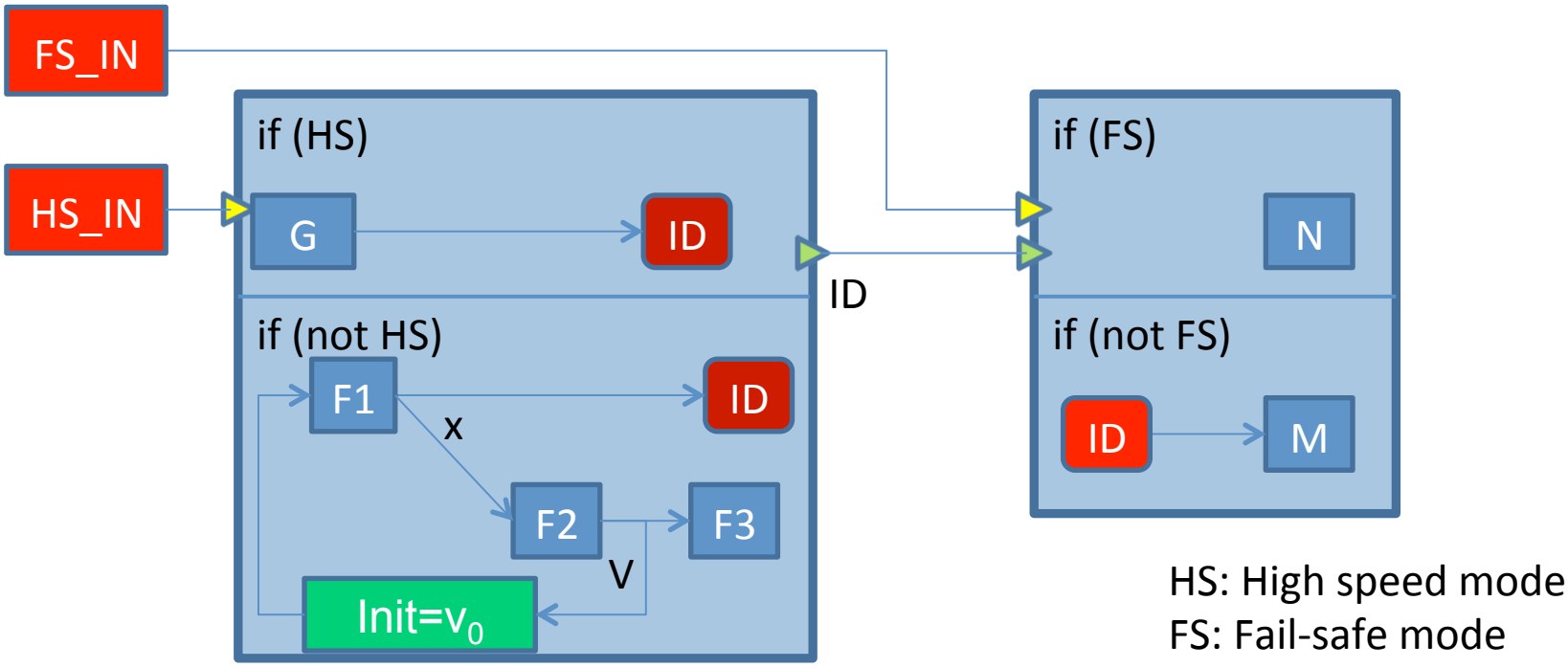
Lopht

Principles of our approach

- Off-line mapping
  - All scheduling decisions are taken before execution
    - **Conditional execution allowed**
  - Construction of **scheduling tables**
- Take into account both computations and communications during scheduling
  - Durations (WCETs, WCCT/WCTTs)
  - Various communication media
    - Buses, shared RAMs, complex networks (NoCs)
    - Each requires different communication and synchronization protocols
      - Differences in scheduling algorithms and code generation
      - Tools: Lopht, SynDEx

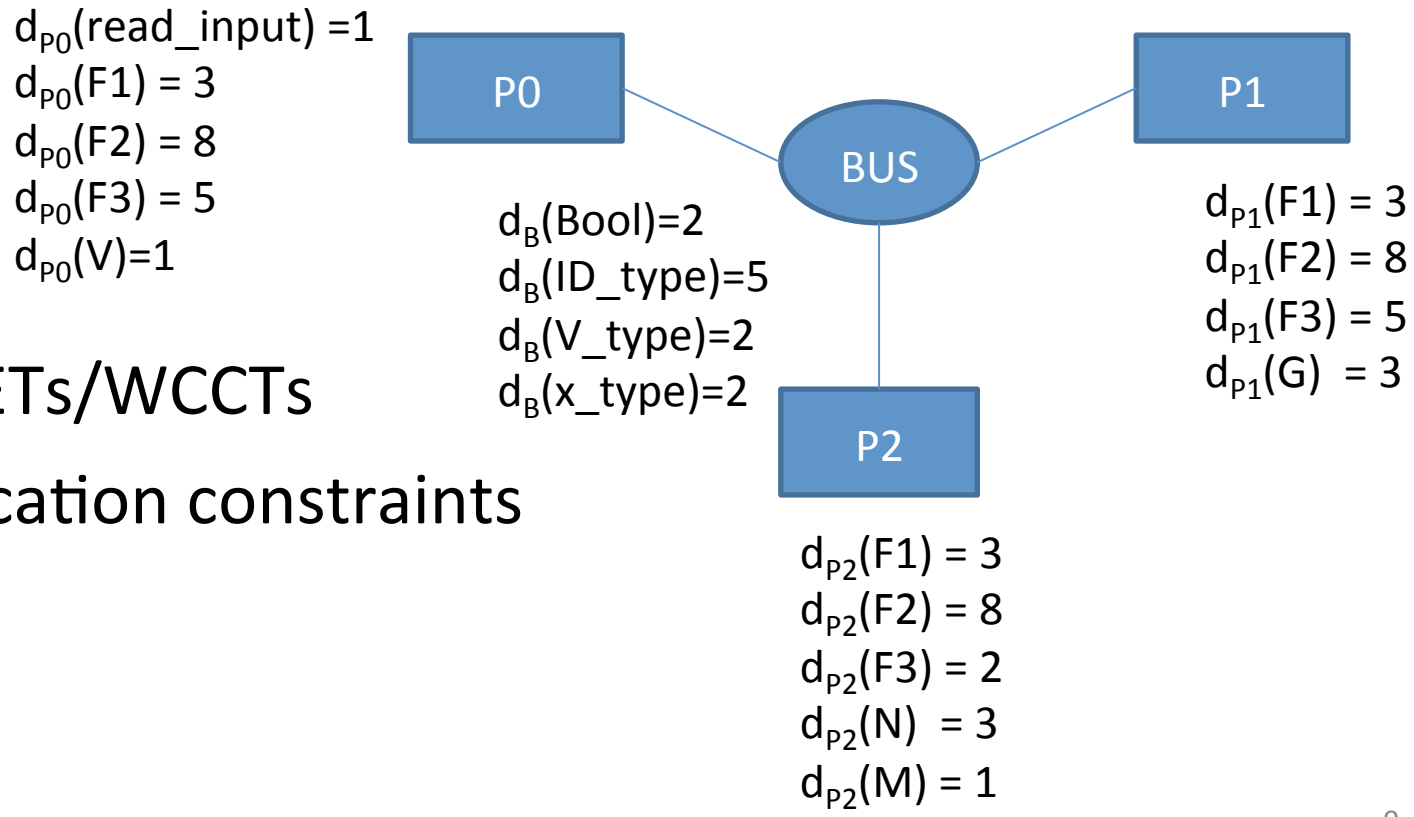
Principles of our approach

- Off-line mapping: ignition engine controller



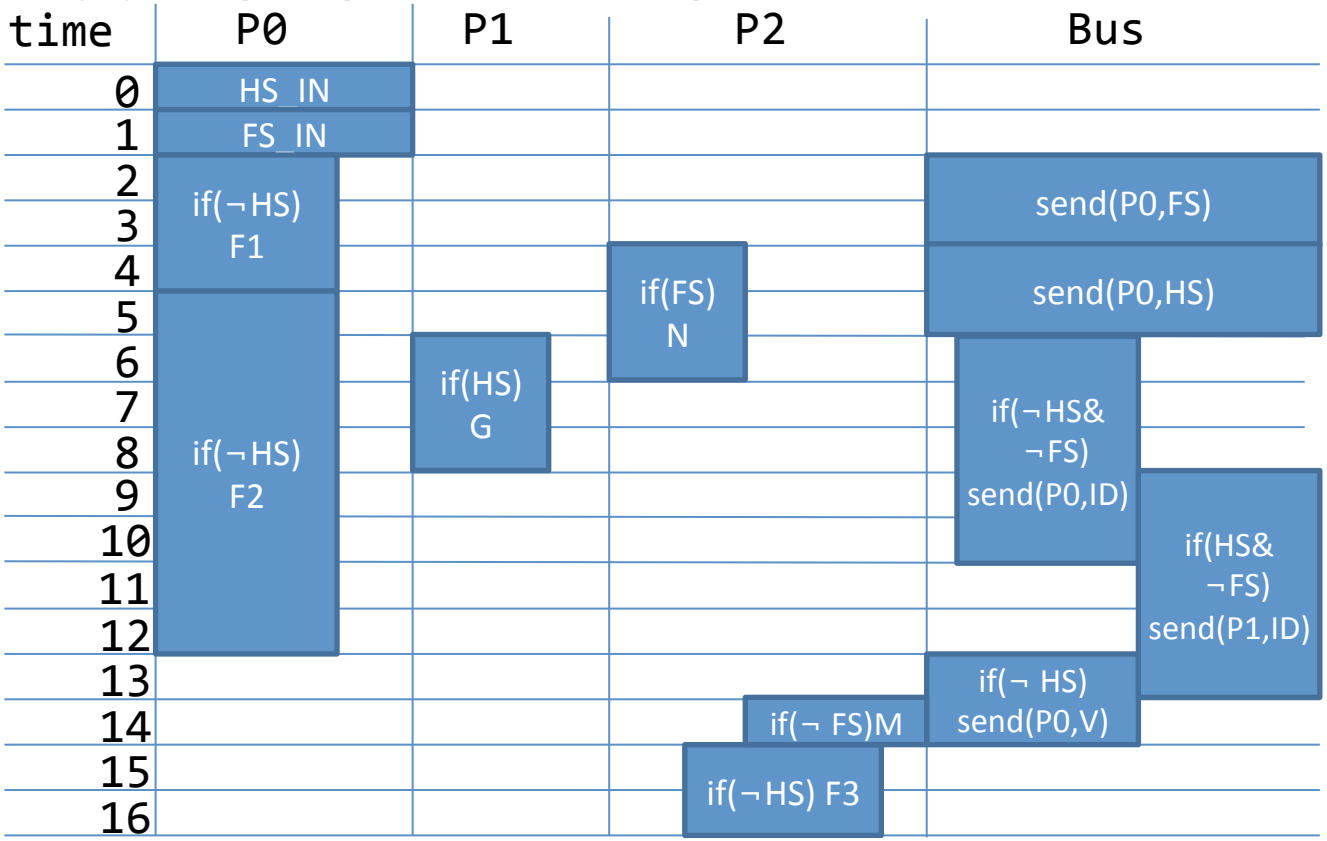
Principles of our approach

- Off-line mapping: ignition engine controller
  - 3 CPUs, 1 broadcast bus
  - WCETs/WCCTs
  - Allocation constraints



Principles of our approach

- Off-line mapping: ignition engine controller



Principles of our approach

- Compilation-like aproach
  - Reliance on fast, efficient heuristics
    - List scheduling combined with
      - Software pipelining to improve throughput/schedulability
      - Deadline-driven scheduling to improve schedulability
      - Post-scheduling optimizations: to reduce the number of context/ partition switches, reduce the number of tasks
    - Easy to take into account:
      - Execution conditions/modes in the functional specification
      - Complex non-functional properties
      - « OS costs » (synchronization, preemption, partition changes, driver cost)
      - Interferences due to access to shared resources (e.g. RAM)
    - Easy to report errors and integrate in an interactive approach

Airbus DS Case study

- Complex application:
  - 13 tasks, 7 partitions, 10 end-to-end flow constraints
  - Complex non-functional requirements: Partitioning, Real time (release dates, flow latencies), Allocation
- Initial architecture:
  - 4 processors, 1 broadcast bus, global time base
- Result
  - Processor charge: 82%, 72%, 72% and 10% (telemetry)
  - Bus charged at 81%

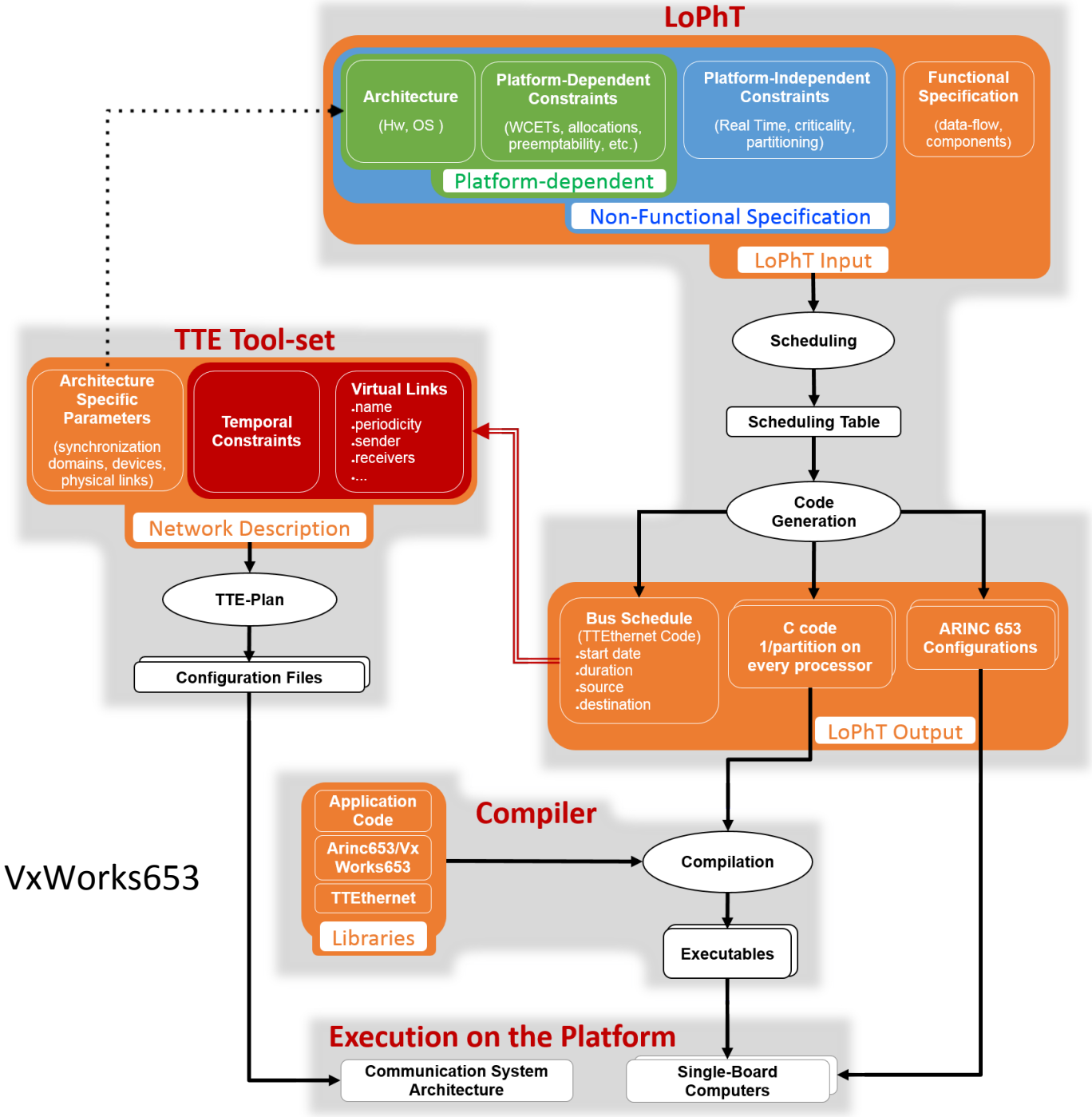
- Now, we need to consider a real, industrial time-triggered communication infrastructure ensuring the global time base
  - **TTEthernet**

TTEthernet

- Communication network (not a bus)
  - Unidirectional links connecting end-stations and switches
    - **Each link has a separate arbiter/scheduler**
  - Store-and-forward packet switching policy
  - All traffic organized in virtual links (VLs) with fixed route and transmission policy
    - Time-triggered (TT) – strictly periodic, one packet per VL period
    - Rate-controlled (RC) – AFDX-like
      - Include clock synchronization traffic (protocol control frames = PCF)
    - Best effort (BE)
- Global time reference, 3 time scales
  - Raster tick – all dates and deadlines are formulated in raster ticks
    - At most one TT packet/raster tick (minimum_delta_r = 1)
    - Recommended: 200usec if network speed is 100mbps
  - Integration Cycle – Periodicity of PCFs
    - multiple of raster tick
  - Cluster Cycle – Length of scheduling table and sync with VxWorks 653
    - multiple of the Integration cycle and of the period of all TT VLs

Lopht for TTEthernet

- Objective: joint mapping of tasks and TTEthernet communications
  - Choose VL routes
  - Choose TT packet scheduling for each link
- Challenges:
  - Build a global scheduling table including computations on processors and TT packets on links
  - Control interferences from non-TT traffic
  - Interoperation with TTE configuration tool (TTE-Plan)


LoPhT
Architecture
(Hw, OS)
Platform-Dependent Constraints
(WCETs, allocations, preemptability, etc.)
Platform-dependent
Platform-Independent Constraints
(Real Time, criticality, partitioning)
Functional Specification
(data-flow, components)
Non-Functional Specification
LoPhT Input
Scheduling
Scheduling Table
Code Generation
TTE Tool-set
Architecture Specific Parameters
(synchronization domains, devices, physical links)
Temporal Constraints
Virtual Links
.name
.periodicity
.sender
.receivers
....
Network Description
TTE-Plan
Configuration Files
Bus Schedule
(TTEthernet Code)
.start date
.duration
.source
.destination
C code
1/partition on every processor
ARINC 653 Configurations
LoPhT Output
VxWorks653
Application Code
Arinc653/Vx Works653
TTEthernet
Libraries
Compiler
Compilation
Executables
Execution on the Platform
Communication System Architecture
Single-Board Computers

Lopht/TTE-Plan interoperation

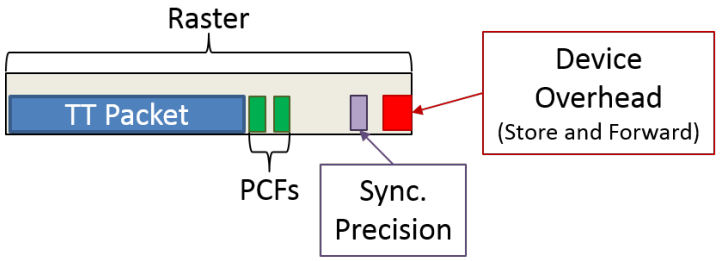
- TTE-Plan
  - Input: VL requirements
    - VL type
    - Start and destination systems
    - Real-time requirements
  - Output
    - VL routes
    - Link scheduler configurations
- **Overlapping with Lopht functions**
  - Forget information when building TTE-Plan input
    - Problem if TTE-Plan cannot find a solution afterwards (as it exists)
    - Possible solutions
      - Enrich TTE-Plan input to allow specification of routing and scheduling constraints
      - Completely bypassTTE-plan ?

Control interferences from non-TT traffic

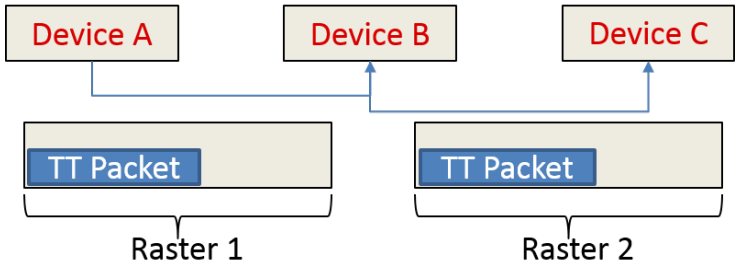
- Hypotheses in TTEthernet use
  - Only TT data traffic
    - Remaining non-TT traffic: clock synchronization (PCF frames of RC type)
  - A TT packet must fit inside a single raster tick
    - 100mbps network => max 120usec/packet (1499-byte payload)
    - Usual raster tick = 200usec

Control interferences from non-TT traffic

- PCF traffic and sync. precision: can be neglected
  - TT packet (120us),
  - PCF (5us),
  - Raster (200us)
  - Device induced delay and network sync precision assumed included in raster



- Multi-hop => overhead of at least one raster tick per link



Build global scheduling table

- Ressources = processors, TTE links
  - Only schedule tasks and TT traffic
- Extension of algorithms for mapping onto network-on-chips
  - Strict periodicity of communications
    - Start with single-period systems
  - Take into account the VxWorks 653/TTE interface
    - Use of queuing pseudo-ports and the critical frame output buffer of the TTE network card of each end system

Conclusion

- Compilaton-like approach in the construction of time-triggered real-time systems
  - Lopht tool working on complex IMA/ARINC 653-based systems (and other types of systems)

- Lopht extension to include TTEthernet networks
  - Already done :
    - Build a high-level model for use by the scheduling algorithms
      - Assumptions on platform use
    - Manual building of a case-study
  - Still to do:
    - Experimentation on the TTE platform
    - Extending the platform modeling language of LoPhT
    - Modifying scheduling and code generation algorithms
    - Tool integration