European Space Agency

Centre For Research & Technology Hellas (CERTH)

Schedulability Analysis Techniques and Tools for Cached and Multicore Processors (MoSATT-CMP)

Final Presentation of ESA Contract No. 4000111814/14/NL/MH
ESA/ESTEC, December 6, 2016

ESA/ESTEC - Dec. 6, 2016

Panagiotis Katsaros
Centre for Research & Technology Hellas (GR)

Multicore Embedded Systems

- Integration of more software functions onto a single platform, to reduce:
  - size and weight
  - cost
  - power consumption

Multicore Embedded Systems

- Integration of more software functions onto a single platform, to reduce:
  - size and weight
  - cost
  - power consumption
- BUT
  - hardware resources shared between concurrent tasks with (possibly) different safety requirements
  - need to ensure predictable timing behaviour through proper schedulability analysis techniques

Design problems

- Spontaneous parasitic timing delays due to bandwidth interference
  - conflicts in simultaneous accesses to shared hardware resources (FPU's, DMA channels, IO peripherals)

Design problems

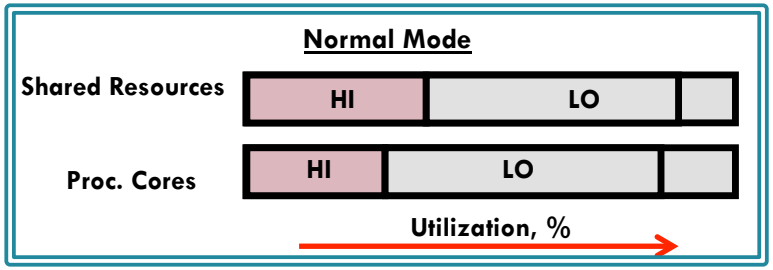
- Spontaneous parasitic timing delays due to bandwidth interference
  - conflicts in simultaneous accesses to shared hardware resources (FPU's, DMA channels, IO peripherals)
- Cache interference
  - additional misses due to sharing (one task modifies the state of the cache memory for another task)

Design problems

- Spontaneous parasitic timing delays due to bandwidth interference
  - conflicts in simultaneous accesses to shared hardware resources (FPU's, DMA channels, IO peripherals)
- Cache interference
  - additional misses due to sharing (one task modifies the state of the cache memory for another task)
- Adaptation to unexpected overload situations (e.g. in autonomous systems)
  - resources (extra margins) to be dynamically reallocated to safety-critical tasks
    - ✓ mixed-criticality scheduling

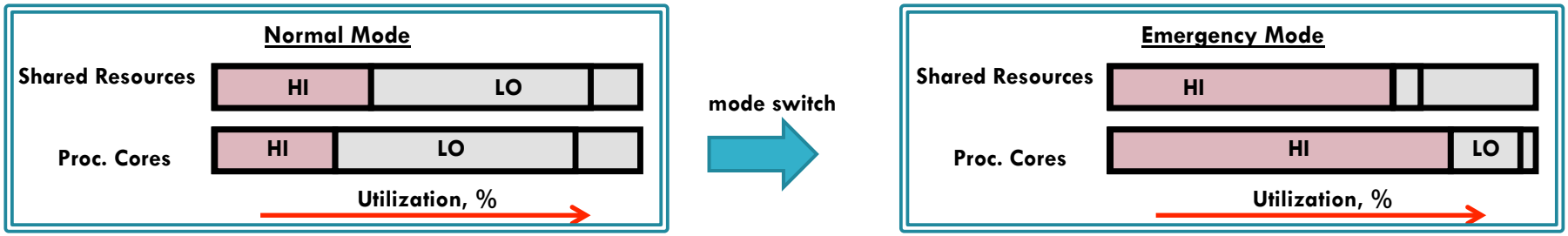
Mixed-criticality scheduling

- A conservative amount of resources is allocated to high-criticality tasks
  - resource budgets not claimed by them in normal operation (no overloads) can be used by the less critical tasks.



Mixed-criticality scheduling

- A conservative amount of resources is allocated to high-criticality tasks
  - resource budgets not claimed by them in normal operation (no overloads) can be used by the less critical tasks.



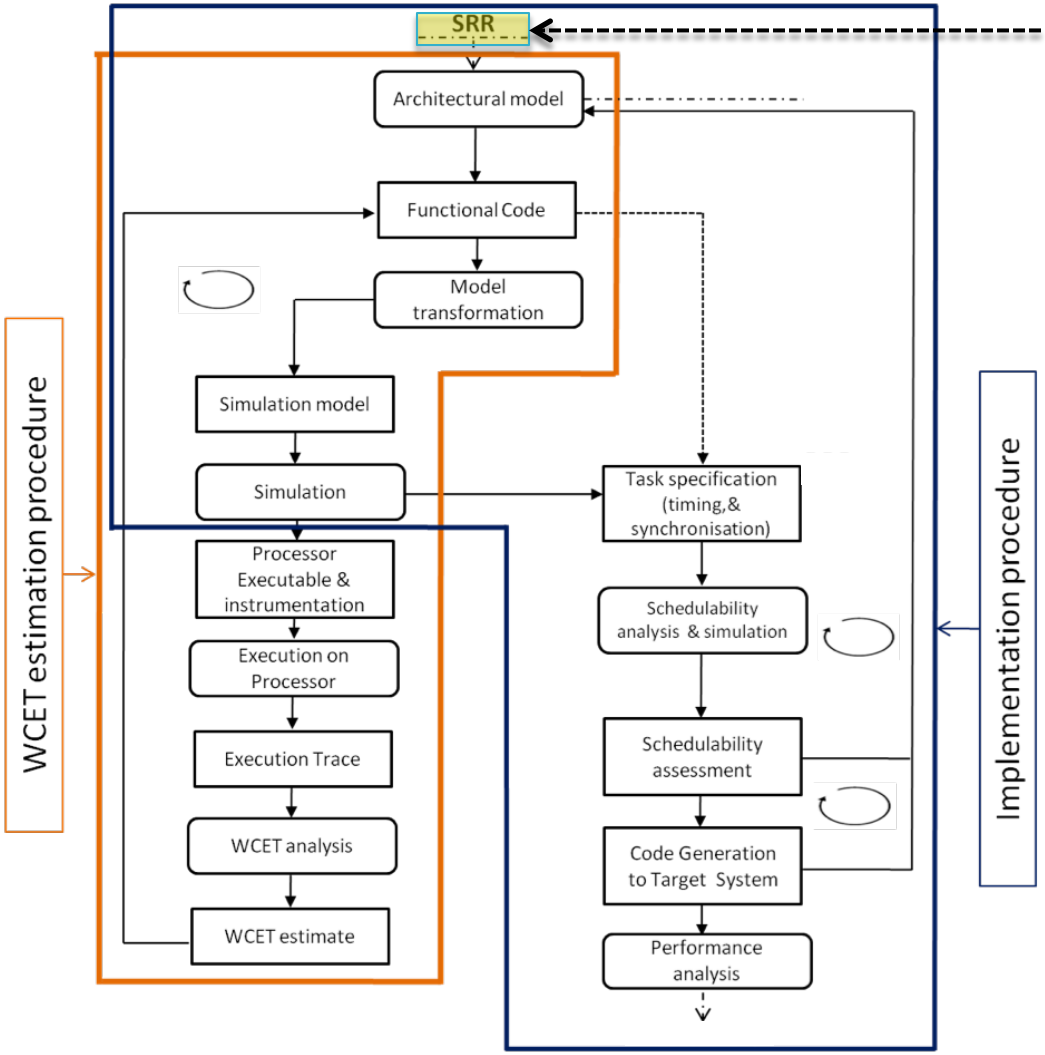
- To free up the resources from LO tasks (mode switch)
  - can be instantaneously aborted and resume later, or
  - in degraded mode (fewer accesses to shared resource)

- Schedulability should be guaranteed, no matter whether and when the mode switch occurs.

MoSATT-CMP design flow

- Not adequate support in Real-Time Operating Systems:
  - for managing the multicore hardware resources
  - scheduling that takes into account interference, as well as mixed-criticality
- Model-based Schedulability Analysis Techniques & Tools for Cached and Multicore Processors: a model-based software design flow for . . .
  - schedulability analysis to ensure the real-time constraints
  - predictable behaviour, through the management of multicore resources

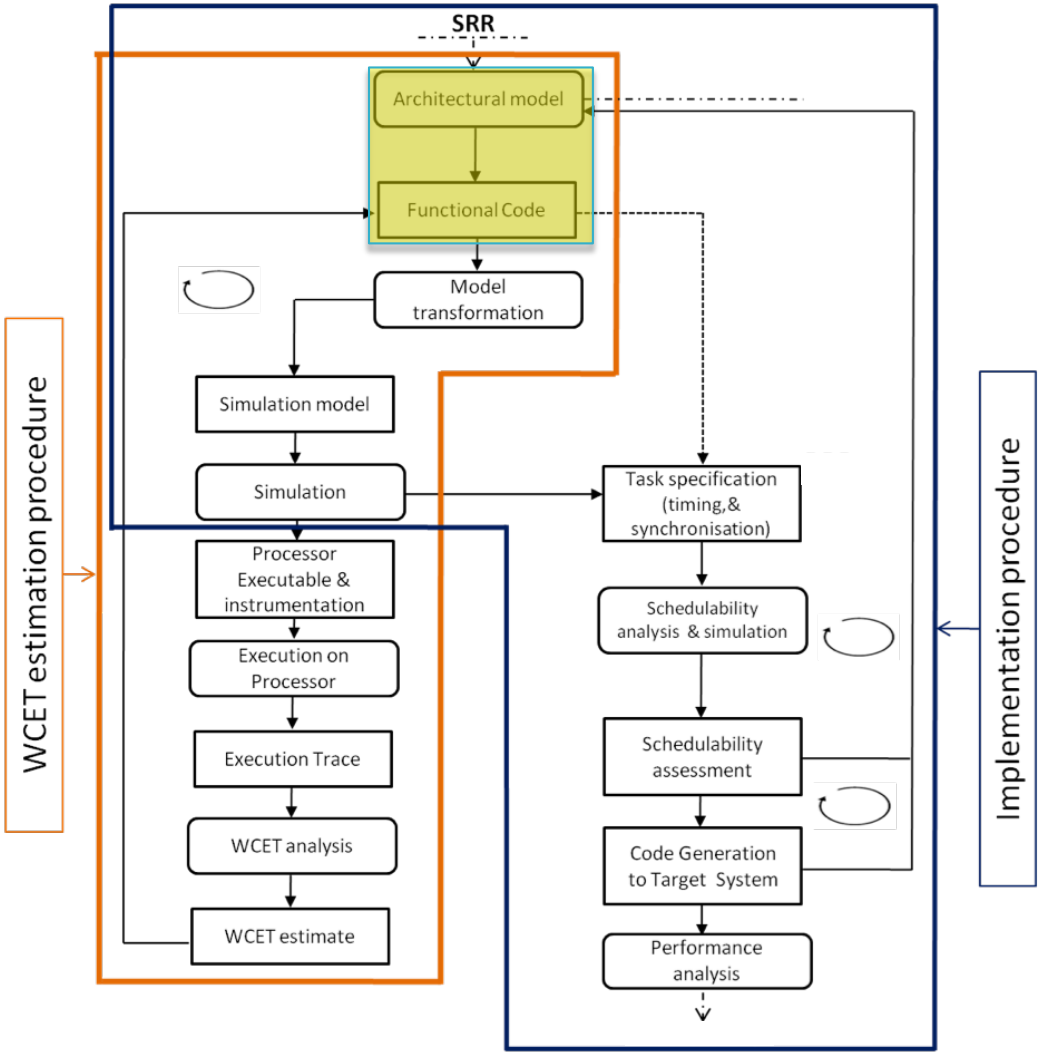
MoSATT-CMP design steps for space systems



Software Requirements Review (SRR)

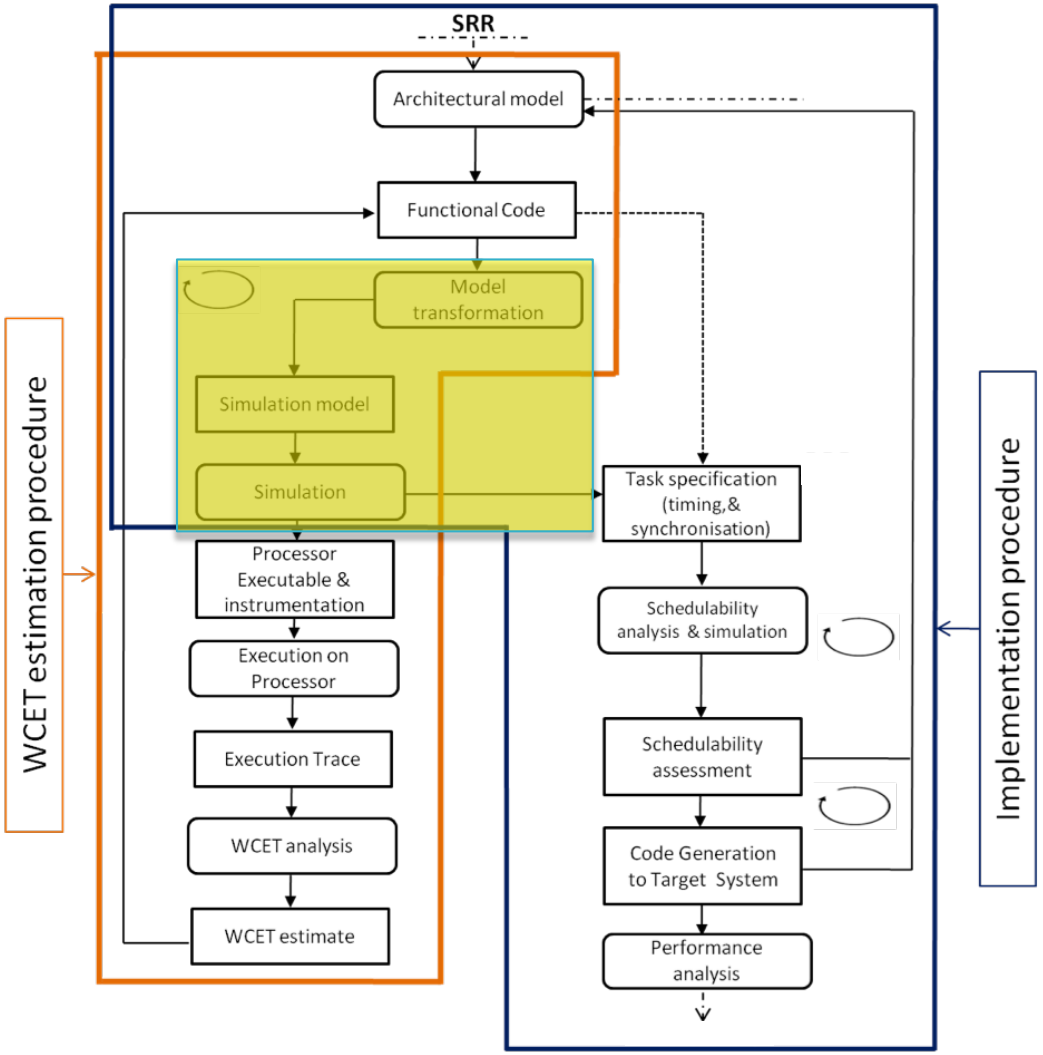
- Resource requirements for software and target hardware
- Requirements for the operation of SW in its environment

MoSATT-CMP design steps for space systems



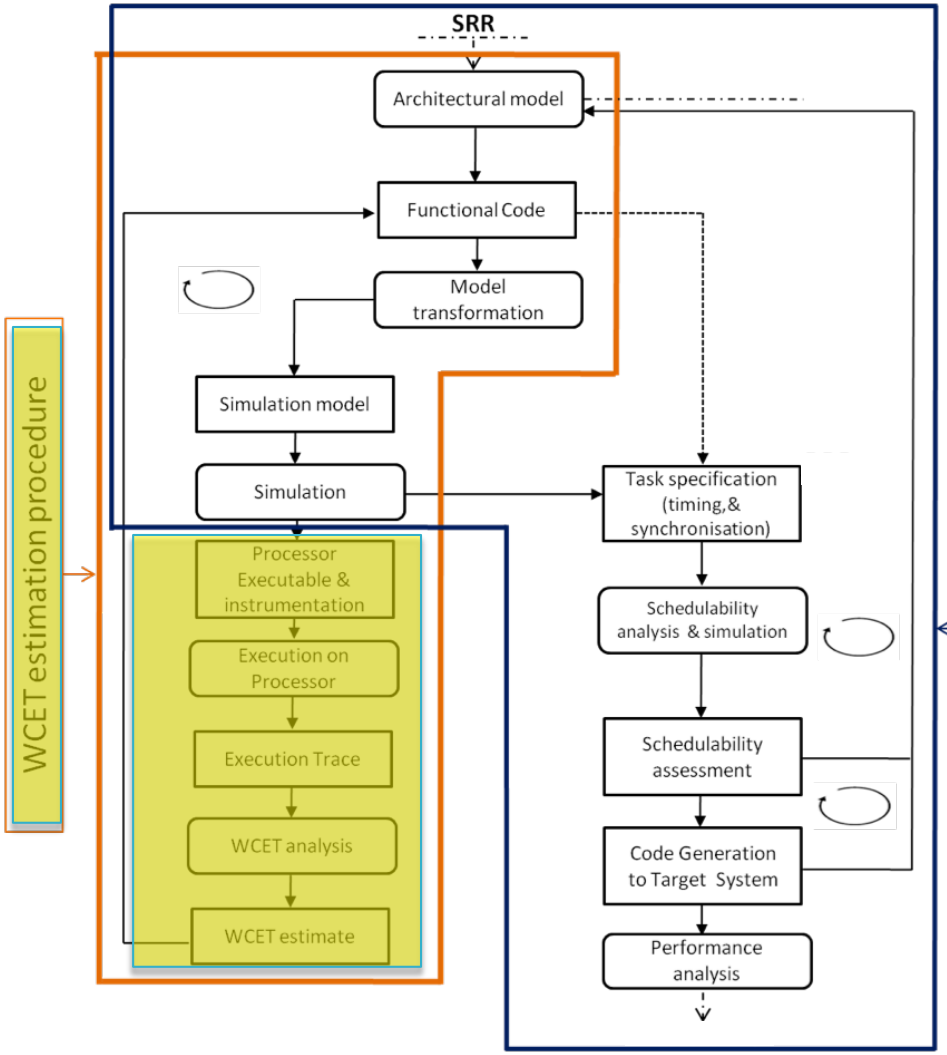
- Requirements are transformed into an architectural model
  - static architecture
  - dynamic architecture
- Functional code (software behaviour) is implemented . . .
- based on an abstract representation of
  - computation, communication & synchronization
  - their temporal & concurrency properties
  - amenable to static analysis

MoSATT-CMP design steps for space systems



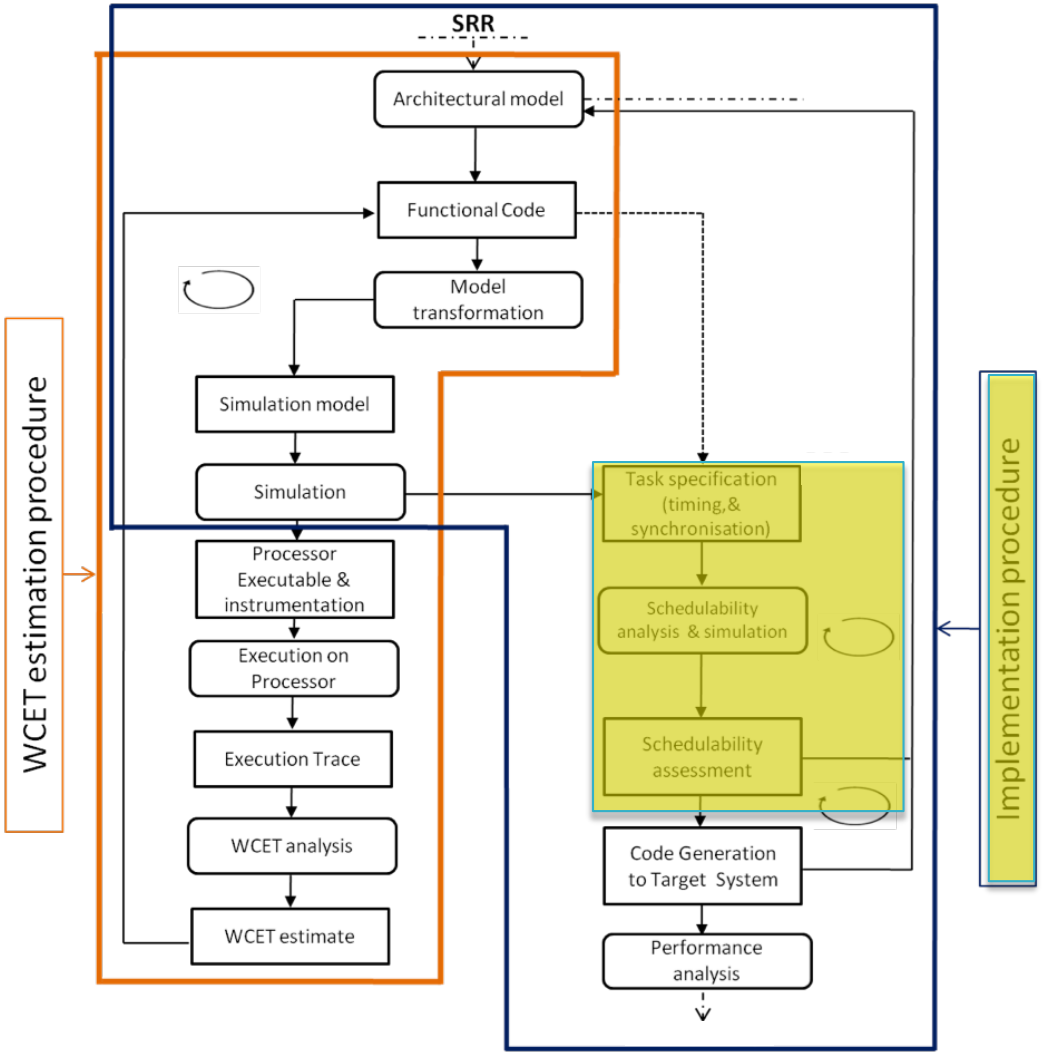
- Automated transformation into a model with formal execution semantics
- Simulation-based validation of the executable model
- . . . allows to refine the design into an implementation that ``fulfills'' the timing constraints

MoSATT-CMP design steps for space systems



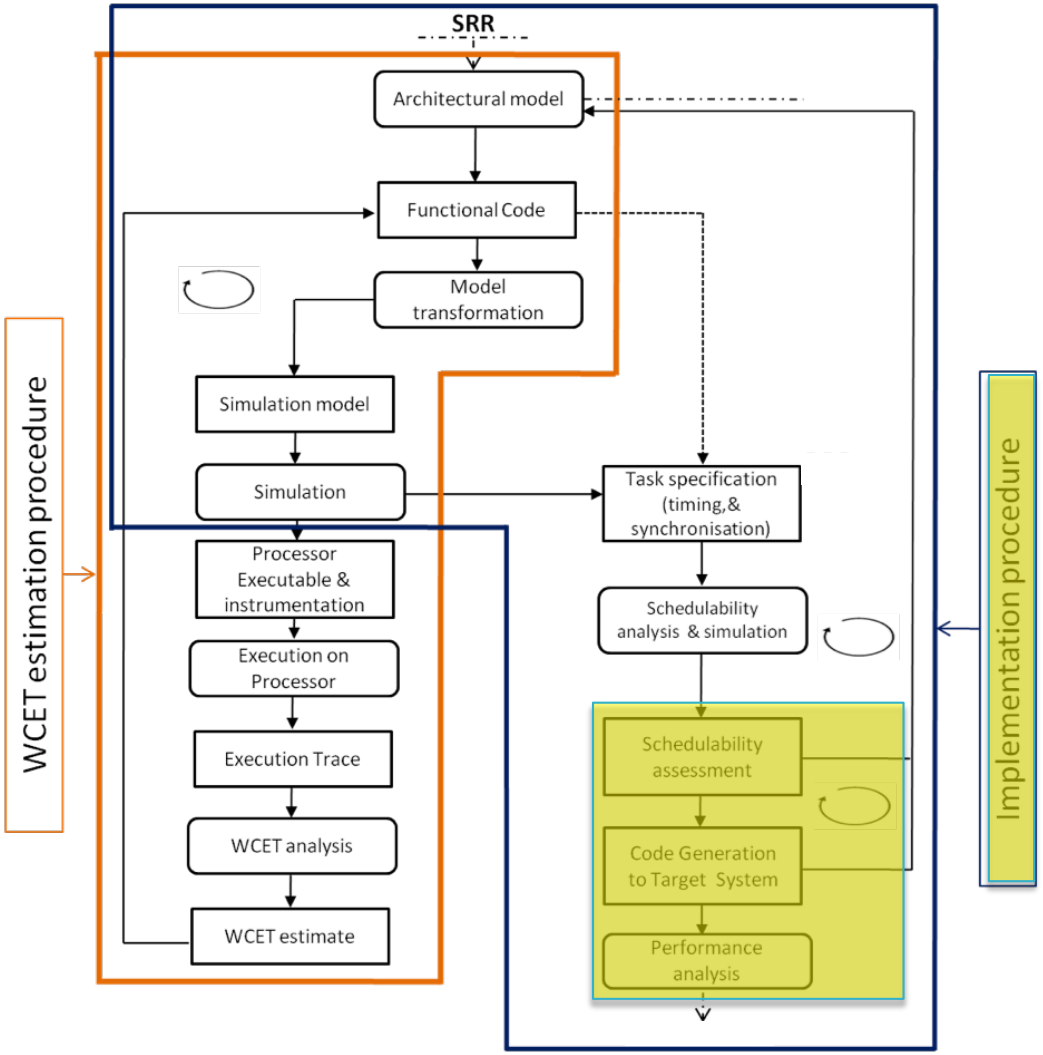
- Worst-Case Execution Time (WCET) estimates for the target hardware platform.
- Measurement-based
- WCET of accesses to shared resources (interference cost)
- . . . fed back into the design model
- Obtain a model with updated timing parameters

MoSATT-CMP design steps for space systems



- Schedulability analysis is based on a unified "concurrency model view" with:
  - all execution entities (tasks, shared vars etc.)
  - SW - HW association
  - scheduling constraints (algorithm, locking for shared variables etc.)
- Guarantee timing behaviour is
  - deterministic or
  - predictable
- Validation of scheduling by simulation or verification

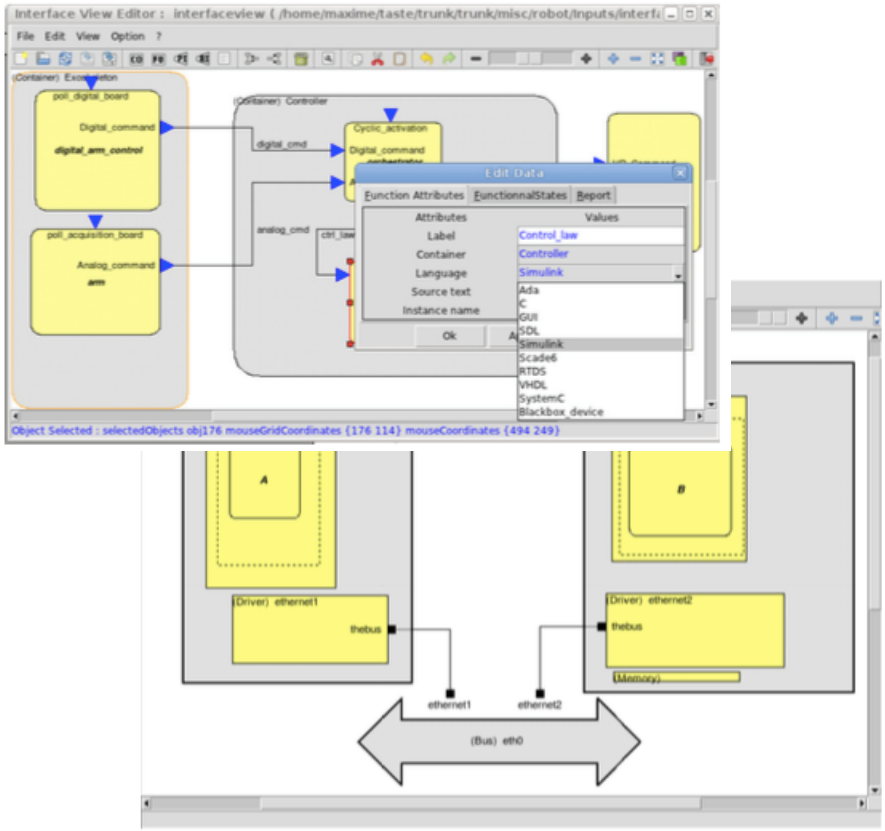
MoSATT-CMP design steps for space systems



- Automated code generation for the target execution platform:
  - user-defined scheduling with minimal run-time support
  - "what you verify is what you execute"
- Possible excessive delays & response times, due to resource starvation cases.
  - validation by tracing/monitoring tools
- performance analysis, if certain certification requirements have to be met

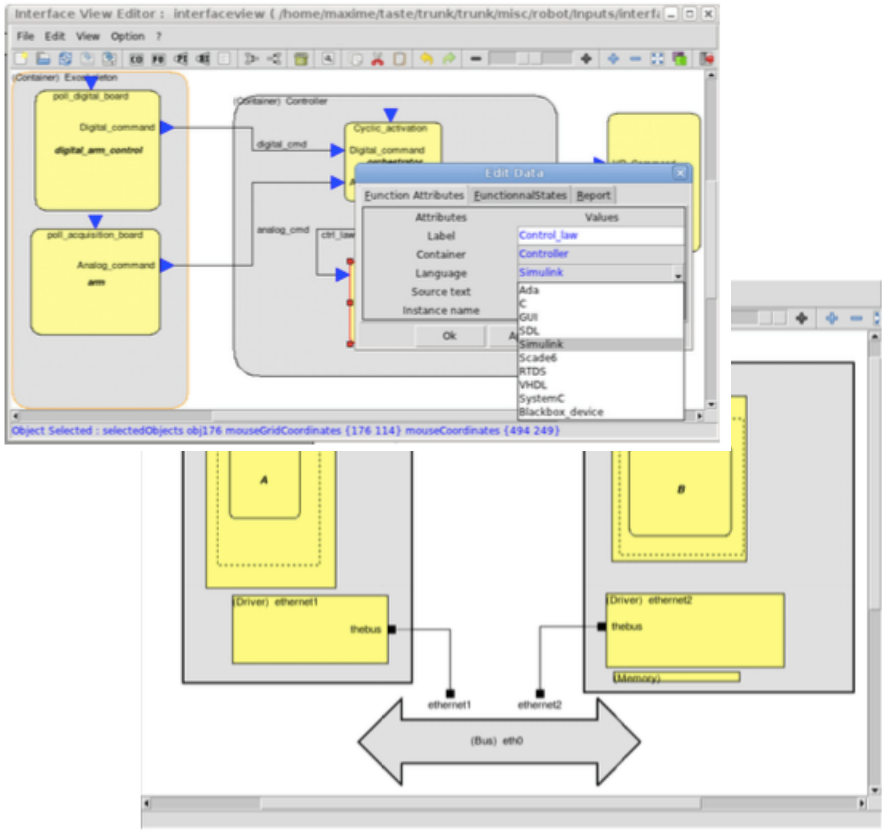
MoSATT-CMP tools are based on TASTE

- Open source tool-chain for model-based design-by-refinement of embedded systems:



MoSATT-CMP tools are based on TASTE

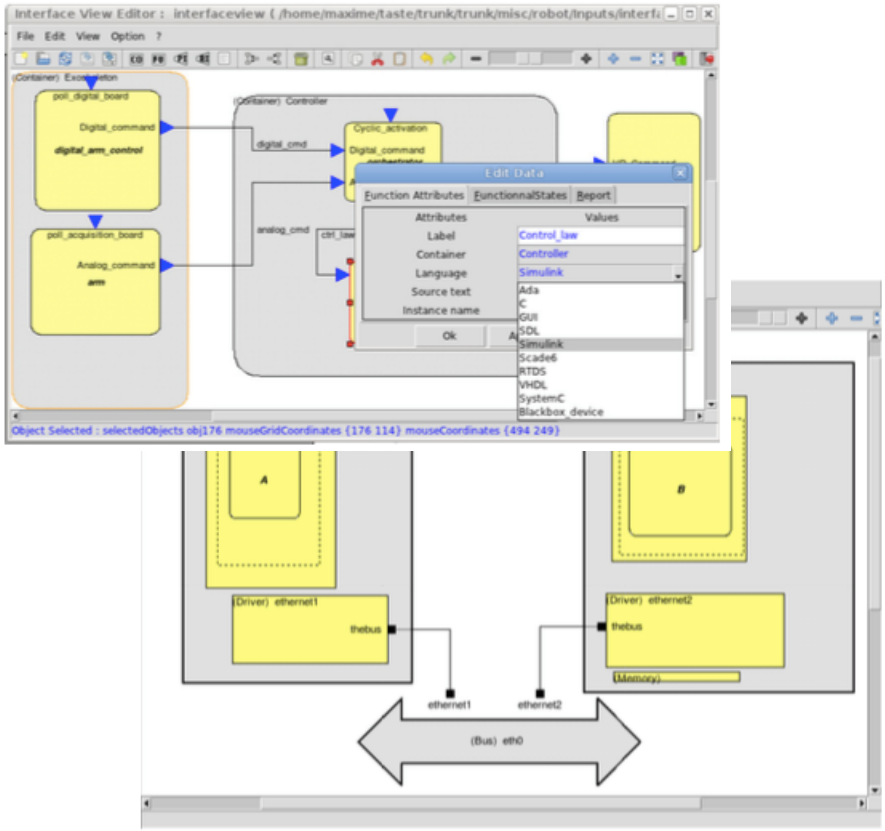
- Open source tool-chain for model-based design-by-refinement of embedded systems:



- testing implementation scenarios derived from a common model

MoSATT-CMP tools are based on TASTE

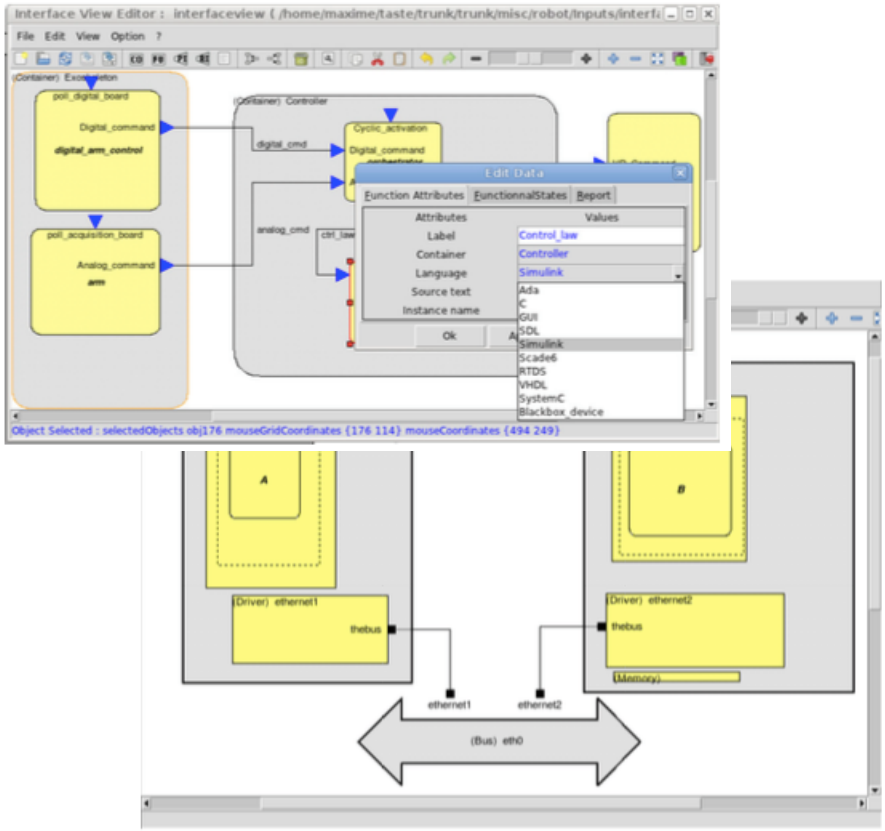
- Open source tool-chain for model-based design-by-refinement of embedded systems:



- testing implementation scenarios derived from a common model
- new scheduling policies via user-defined model attributes

MoSATT-CMP tools are based on TASTE

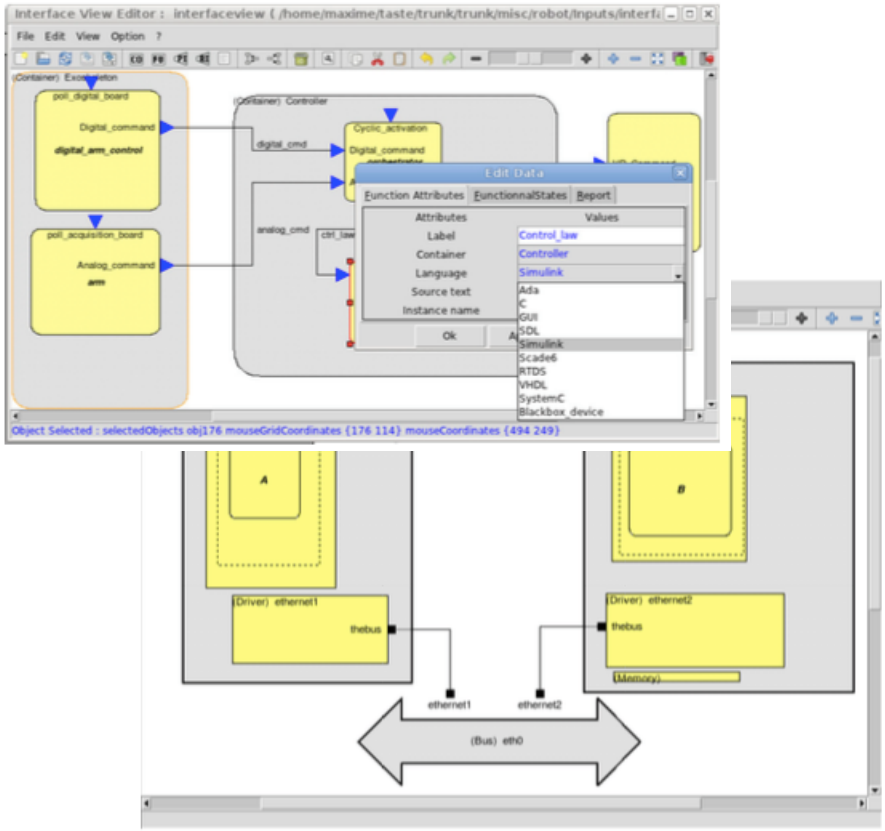
- Open source tool-chain for model-based design-by-refinement of embedded systems:



- testing implementation scenarios derived from a common model
- new scheduling policies via user-defined model attributes
- domain-specific analysis (e.g. model checking, schedulability)

MoSATT-CMP tools are based on TASTE

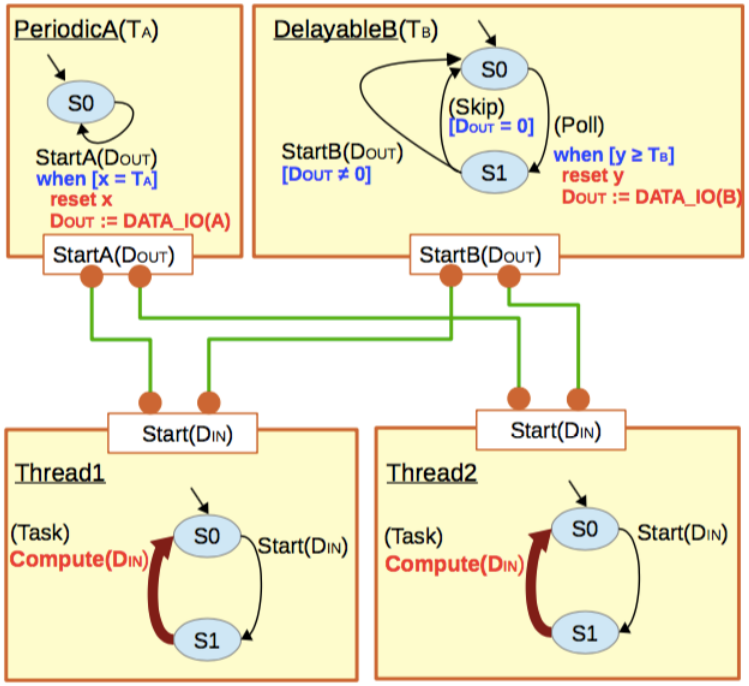
- Open source tool-chain for model-based design-by-refinement of embedded systems:



- testing implementation scenarios derived from a common model
- new scheduling policies via user-defined model attributes
- domain-specific analysis (e.g. model checking, schedulability)
- supports existing languages and tools, appropriate for particular design problems

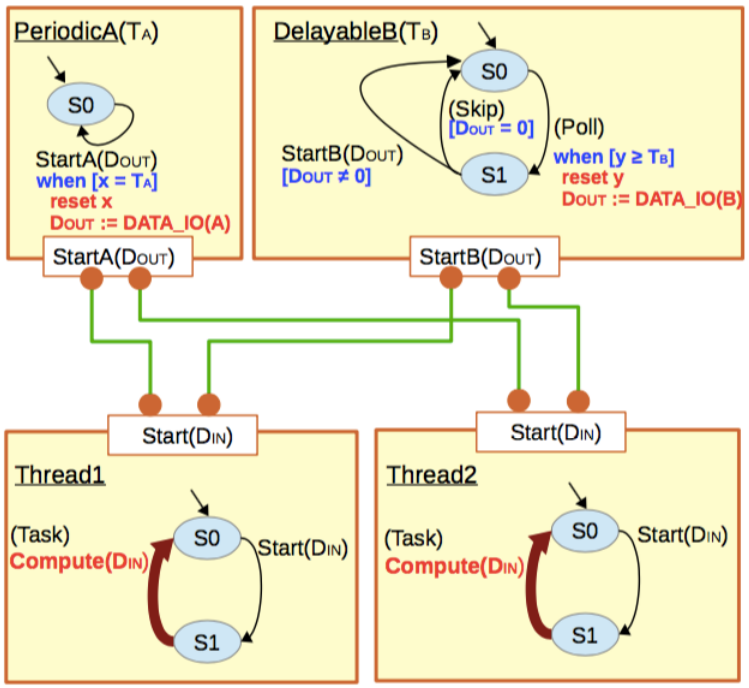
Analysis & code generation using BIP

- RT-BIP formal language: executable models for concurrency & timing behaviour of system software components



Analysis & code generation using BIP

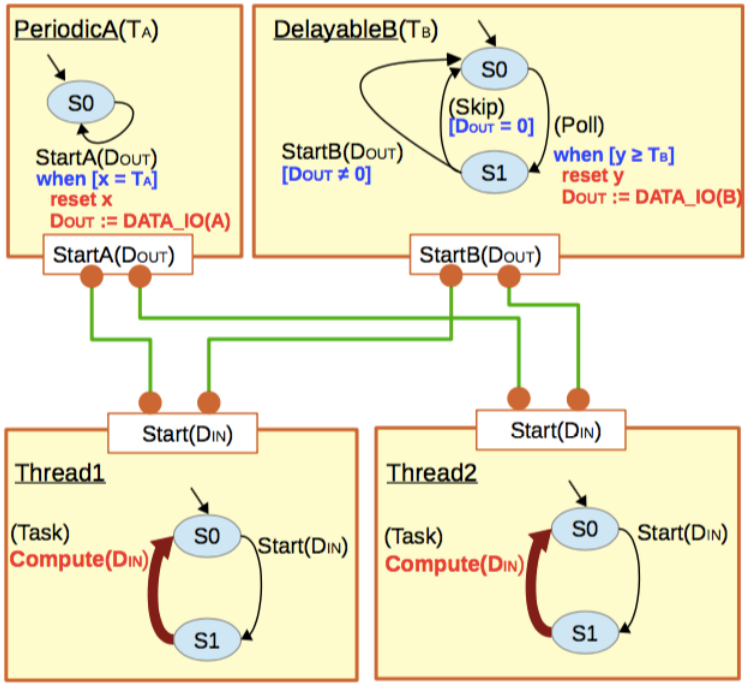
- RT-BIP formal language: executable models for concurrency & timing behaviour of system software components



- interacting task automata (timed automata with transitions that have non-zero execution time)

Analysis & code generation using BIP

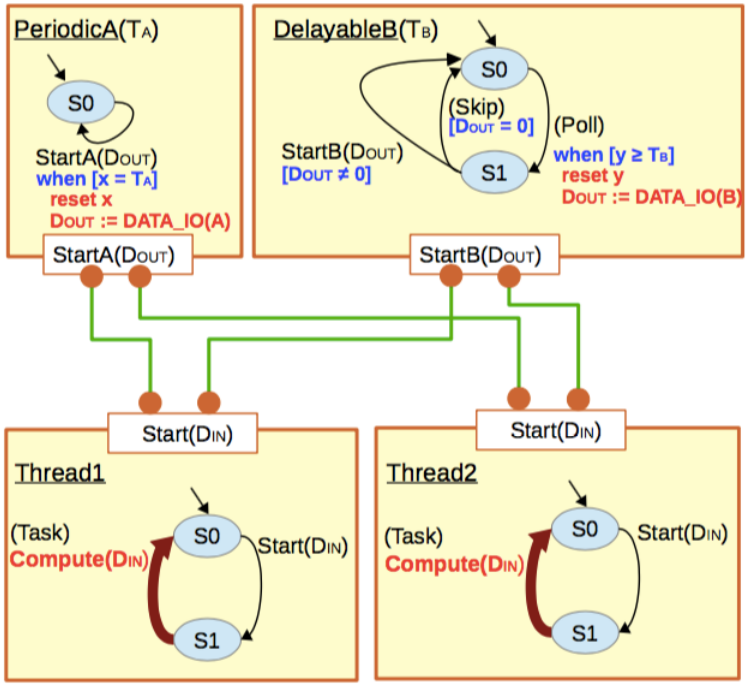
- RT-BIP formal language: executable models for concurrency & timing behaviour of system software components



- interacting task automata (timed automata with transitions that have non-zero execution time)
- BIP model is translated in C++

Analysis & code generation using BIP

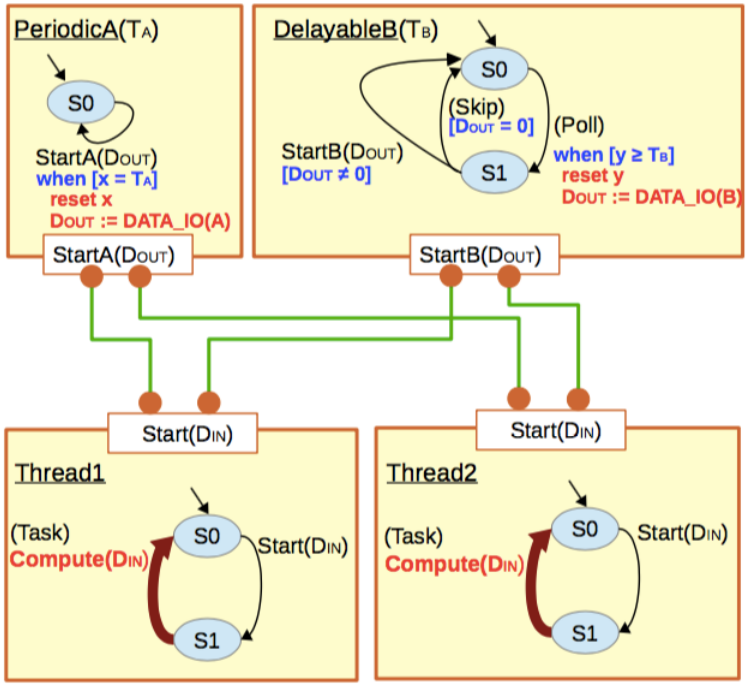
- RT-BIP formal language: executable models for concurrency & timing behaviour of system software components



- interacting task automata (timed automata with transitions that have non-zero execution time)
- BIP model is translated in C++
- linked with the multi-threaded BIP Runtime Environment (RTE)

Analysis & code generation using BIP

- RT-BIP formal language: executable models for concurrency & timing behaviour of system software components



- interacting task automata (timed automata with transitions that have non-zero execution time)
- BIP model is translated in C++
- linked with the multi-threaded BIP Runtime Environment (RTE)
- BIP RTE supports parallel execution of BIP components using POSIX threads

Models of Computation (MoC)

- Technical challenges:
  - schedule tasks while taking into account task dependencies
  - predictable timing behaviour while retaining the efficiency potential through parallel processing
    - ✓ functional determinism

      program's outputs do neither depend on the tasks' execution times nor on the tasks' scheduling

Models of Computation (MoC)

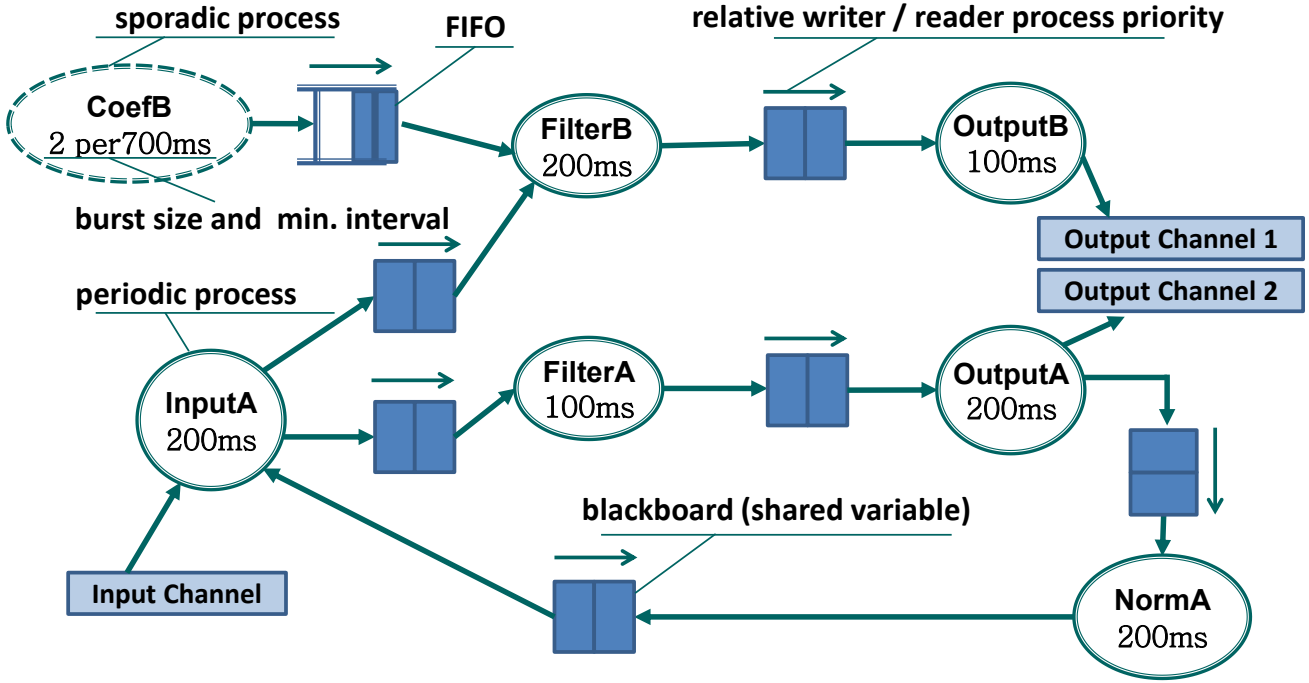
- Technical challenges:
  - schedule tasks while taking into account task dependencies
  - predictable timing behaviour while retaining the efficiency potential through parallel processing
    - ✓ functional determinism

      program's outputs do neither depend on the tasks' execution times nor on the tasks' scheduling

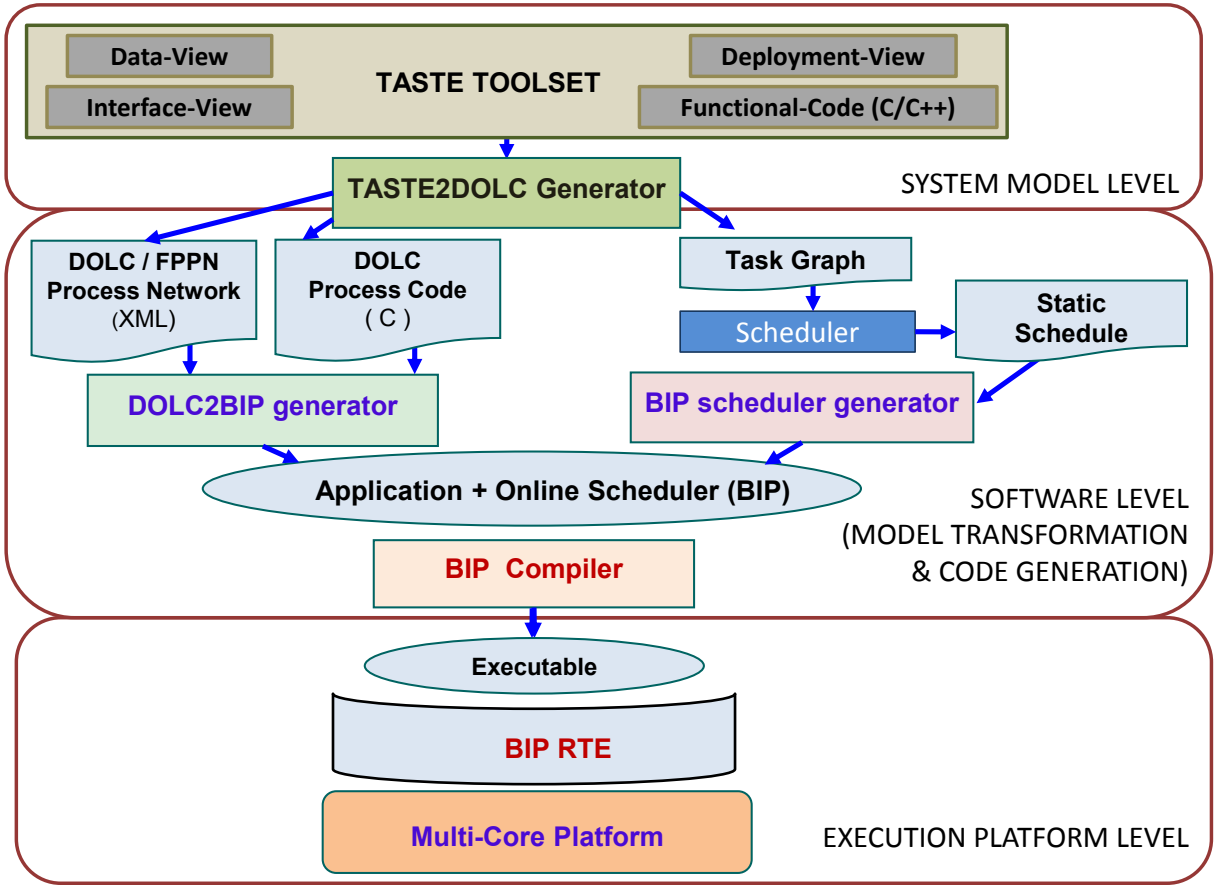
- Adopt a suitable MoC:
  - takes into account the applicable task dependency patterns
  - imposes certain implementation - independent restrictions on the task execution & inter-task communication
  - supports multicore-aware scheduling and analysis techniques

FPPNs: a new process network MoC

- Fixed Priority Process Networks (FPPNs)
  - extends streaming MoC with real-time task properties
  - channels are not necessarily FIFOs
  - supported by multicore-aware schedulability analysis

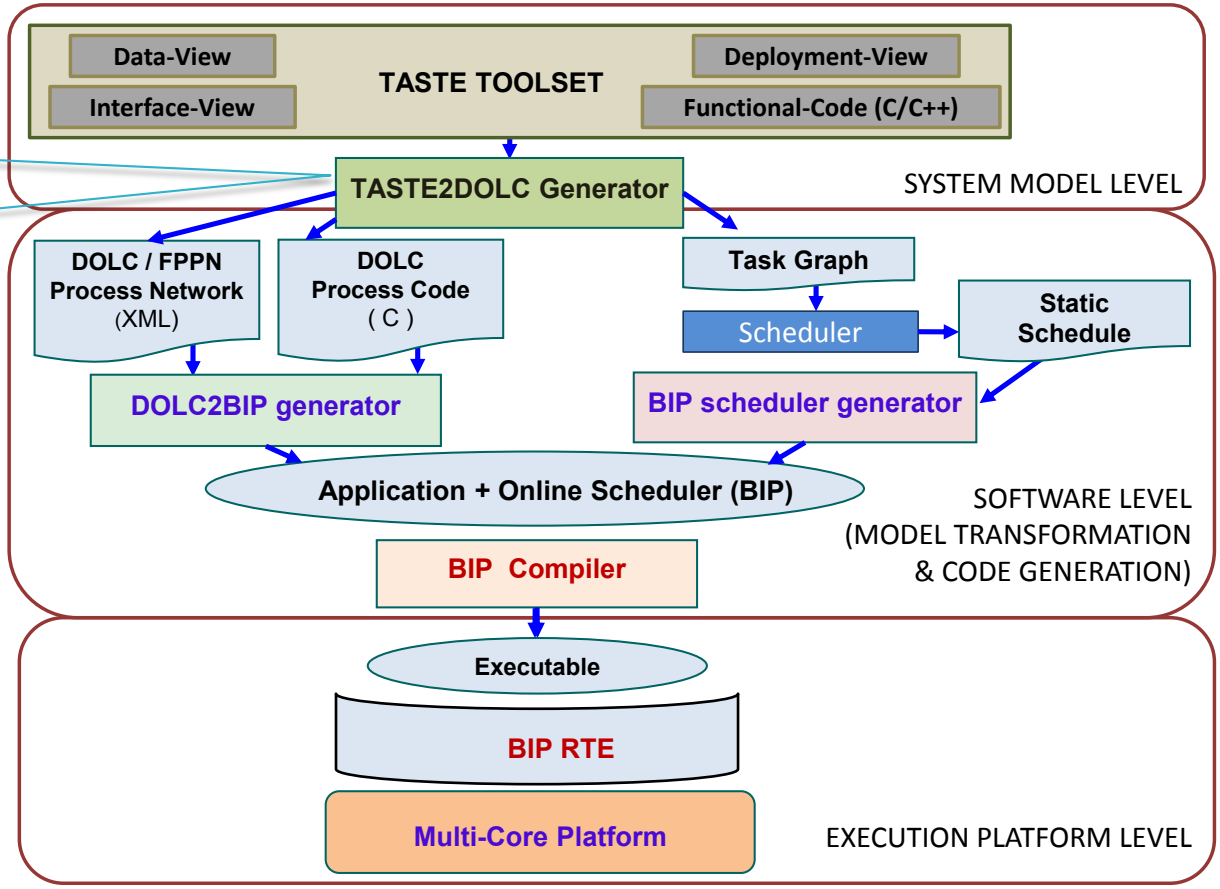


MoSATT-CMP implementation tool-chain



MoSATT-CMP implementation tool-chain

DOLC: an intermediate model representation



MoSATT-CMP WCET estimation I

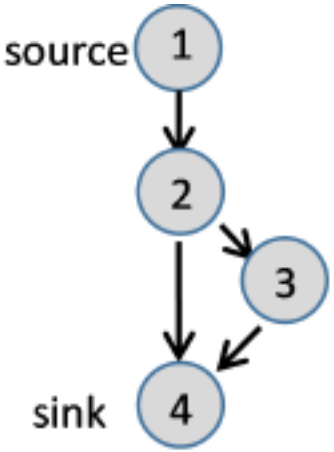
- Measurement-based (relatively easier to implement)
  - application runs in isolation on one core
  - no others tasks are running on any other core
  - measurements not tainted with bandwidth and cache interference from other tasks (worst-case interference depends on the scheduling)
- Measurements:
  - tasks's software parameters – # of times that each code block is executed
  - task's response time
- Code instrumentation using the Rapita Verification Suite (RVS)
  - provides an instrumentation point (IPoint) function
  - trace information from the board is captured by the RTBx data logger for recording time-stamped data (connected via the GPIO port)

MoSATT-CMP WCET estimation II


```
ipoint(1);
i ← i(n)
out ← i*i_1;
i_1 ← i;
ipoint(2);
if (out<0) {
  out ← –out;
  ipoint(3);
}
ipoint(4);
```

β_0 spans ipoint(1) through ipoint(2); β_1 spans if (out<0) { through ipoint(3);

(a) instrumented source code



$$X_1 = f(2,3)$$

Other graph edges removed by the simple elimination procedure.

$$Y(n) \approx \beta_0 + \beta_1 \cdot X_1(n)$$

(b) i-point graph, flow counters and predictors

MoSATT-CMP WCET estimation III

- Highly-probable execution time statistical overestimations
  - avoid the high cost of guaranteeing extremely high probability

$$\mathcal{P}r\{Task\ Execution\ Time < WCET\} > 1 - \alpha$$

(for Extreme Value Theory $\alpha=10^{-15}$, need for independent & identically distributed observations)

MoSATT-CMP WCET estimation III

- Highly-probable execution time statistical overestimations
  - avoid the high cost of guaranteeing extremely high probability

$$\mathcal{P}r\{Task\ Execution\ Time < WCET\} > 1 - \alpha$$

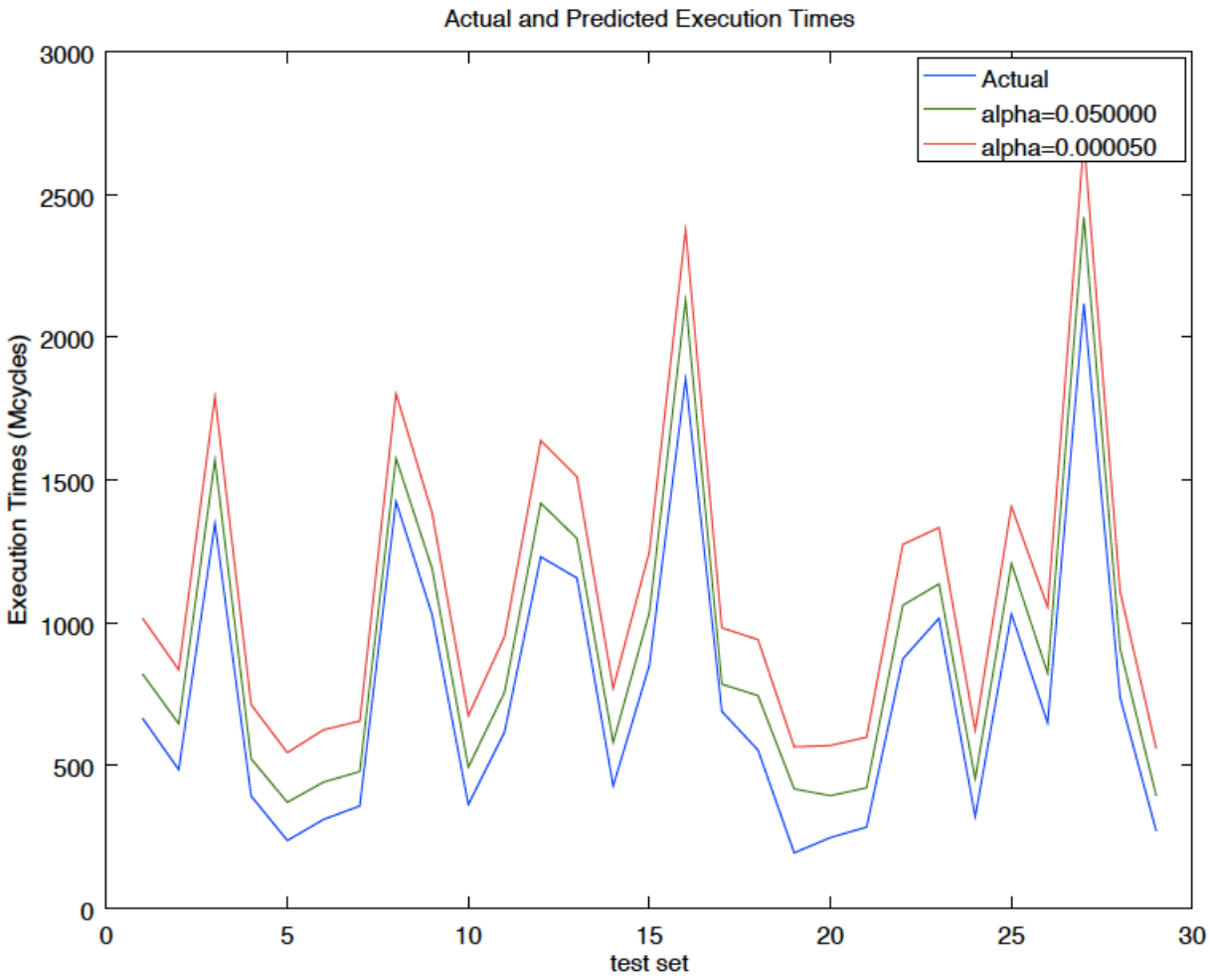
(for Extreme Value Theory $\alpha=10^{-15}$, need for independent & identically distributed observations)

P. Poplavko, L. Angelis, A. Nouri, A. Zerzelidis, S. Bensalem, P. Katsaros. *Regression-based Statistical Bounds on Software Execution Time*. Verimag Research Report no TR-2016-7, Grenoble, France, November 2016

  - cope with the complex dependency on input data
  - find `adequate' dependency model with `random' errors
  - take advantage of the rich set of automated statistical model-fitting tools (stepwise regression fitting)
  - we can assess adequacy (safe and tight overestimations) and randomness with objective statistical indicators

Use case: JPEG Decoder application on Leon 4

- Measurements: 99 images of different sizes and color formats (training set: 70 cases, test set: 29 cases)
- Traces: 389 blocks of code



Cache Interference Analysis for the NGMP

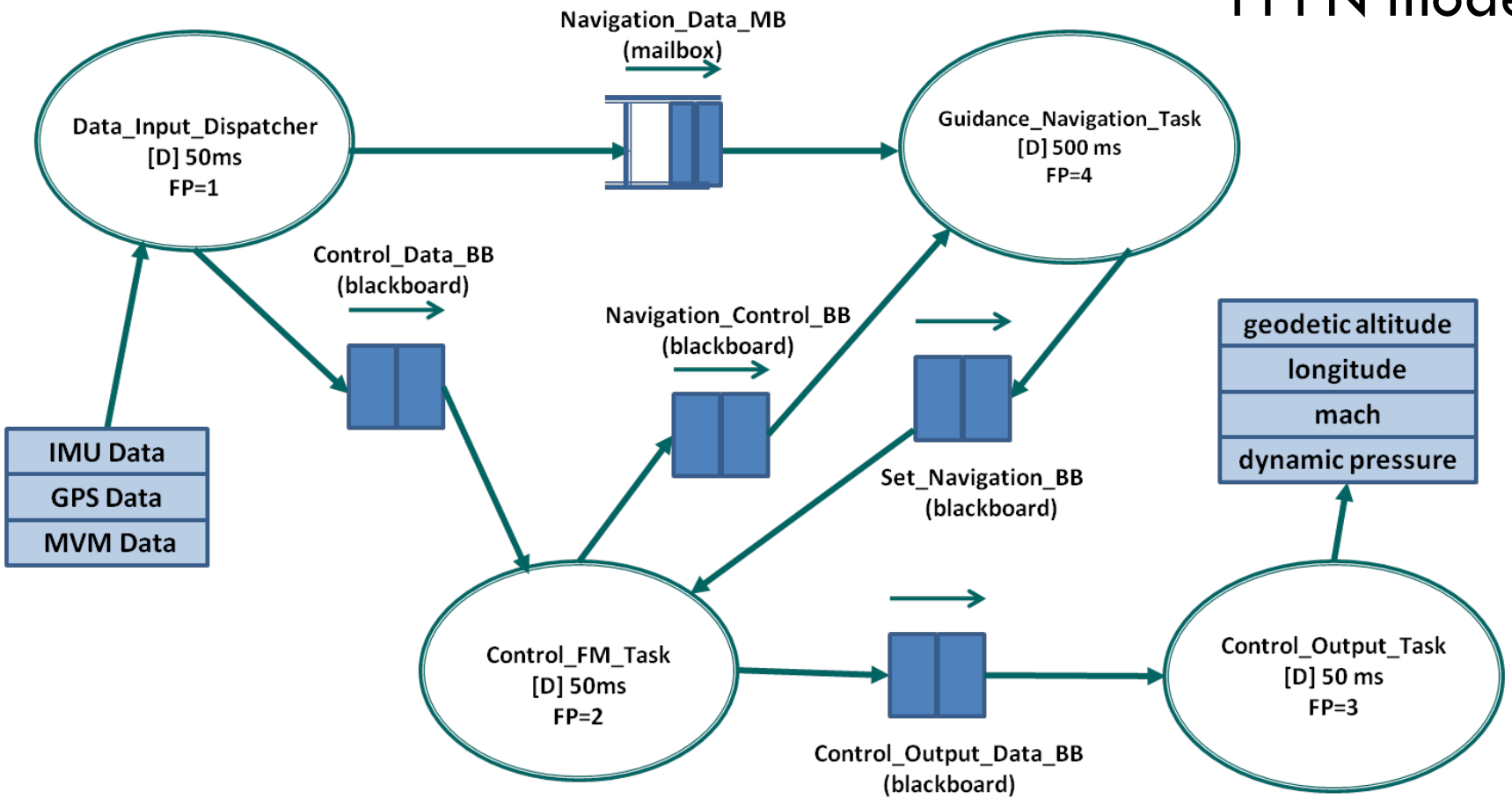
- C program analyzing the memory access trace of an application to determine the latency added by cache misses
  - reuse distance of a memory access to a block: # memory accesses between the current and the previous access to the block
- Memory access trace:
  - gathered from the LEON4-N2X trace buffers via the Debug Support Unit and the GRMON2 debugger
- Issues:
  - program runs a number of instructions until the trace buffers are getting full (procedure automated with a tcl script)
  - Very time consuming: 30 hours for trace with 1,3 million instructions

Use case: GNC app by Elecnor Deimos-Space S.L.U

- Guidance, Navigation & Control application originally built for the Leon3 single-core processor
  - GNC app modelling in TASTE Interface View (TASTE IV)
  - RTEMS calls removed from the C code
  - TASTE IV functional C code primitives generated
  - task graph & functional FPPN/DOLC model produced (TASTE2DOLC tool)
  - BIP model & code for the BIP RTE generated (DOLC2BIP tool)
  - code instrumentation & WCET analysis
  - BIP RTE ported to the Leon4
  - application running on Leon4 under various schedules

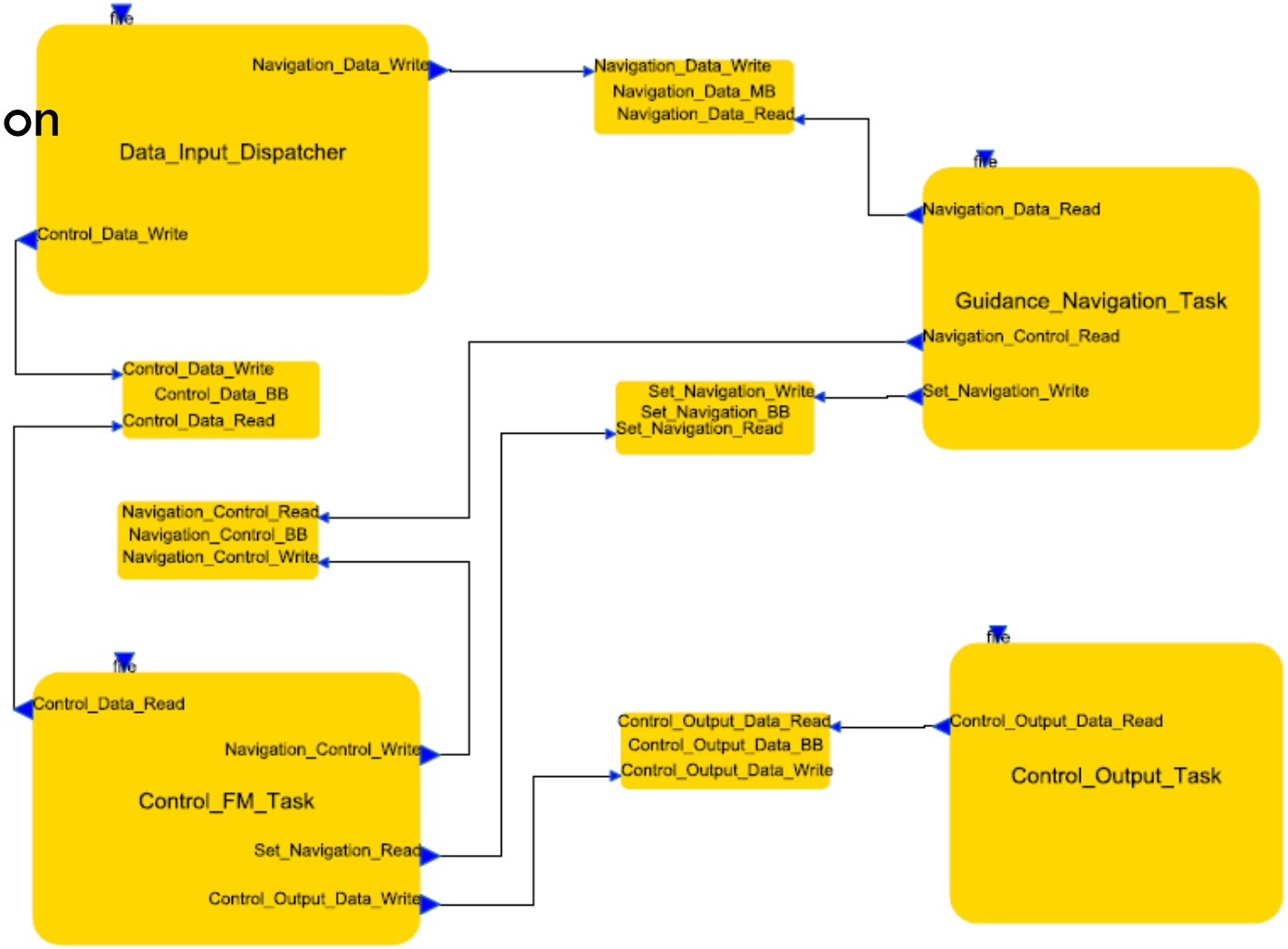
Use case: GNC app by Elecnor Deimos-Space S.L.U

FPPN model



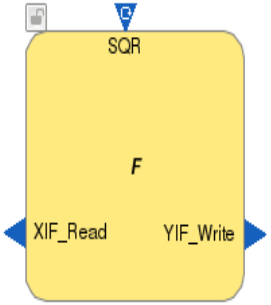
Use case: GNC app by Elecnor Deimos-Space S.L.U

TASTE IV representation



TASTE IV tasks compiled to BIP

provided interface
period *T*



required interfaces

XIF_Read(IN x, IN valid)

YIF_Write(OUT y)

```
void SQR_Init() {
    index = 0;
}

void SQR_Execute() {
      XIF_Read(&x, &x_valid);
      if (x_valid) {
           y  = x * x;
           YIF_Write(&y);
       }
       index = index + 1;
}
```

state

discrete transition

continuous transition

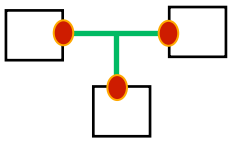
when [*x = T*] timing condition

[*valid*] data condition

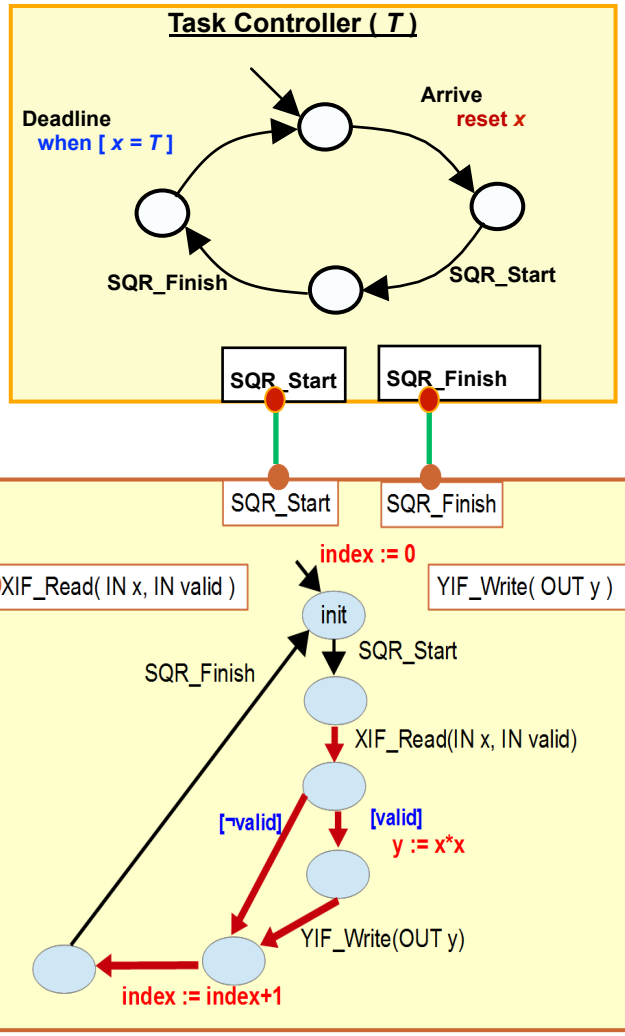
reset *x* timing action

y := x*x data action

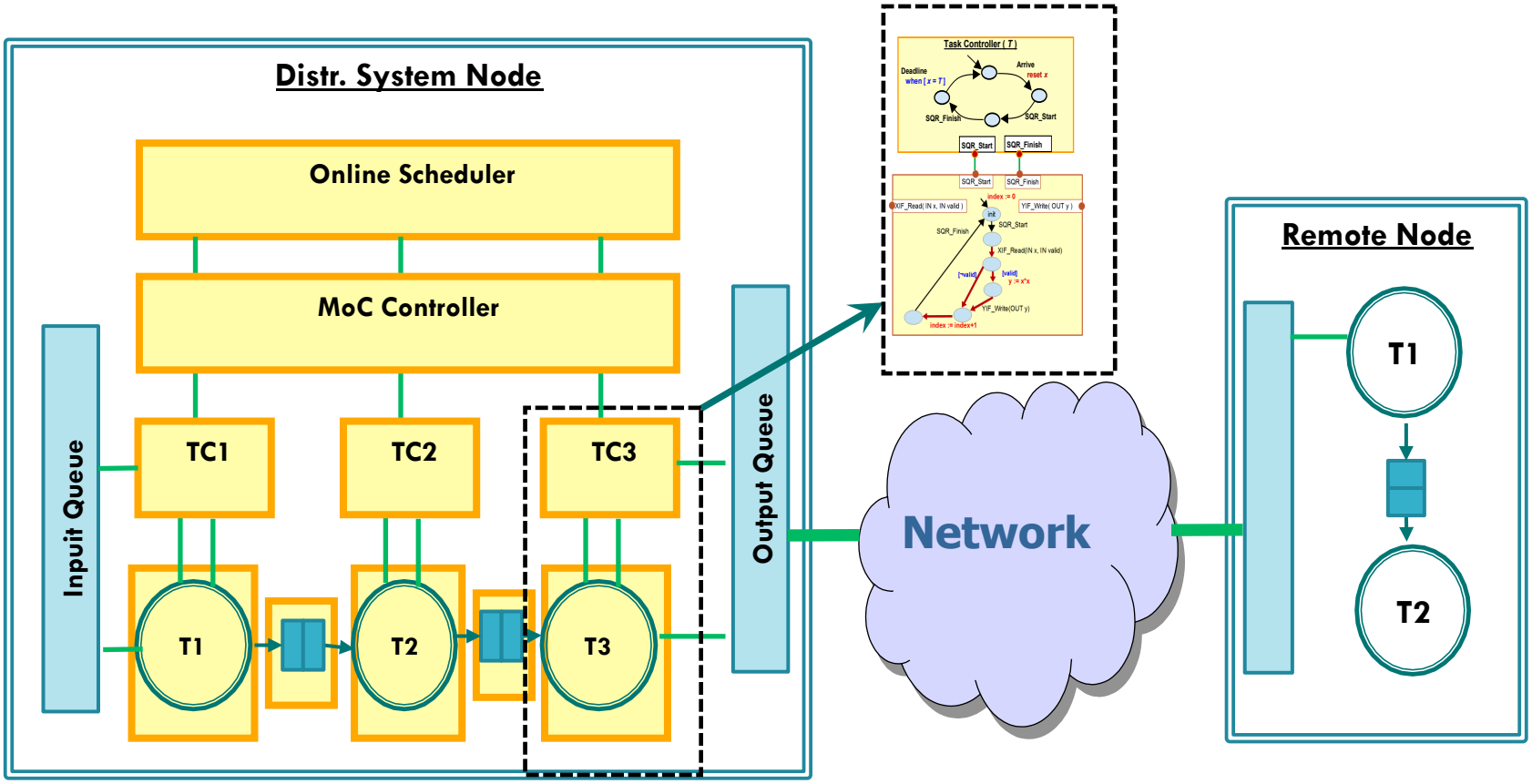
YIF_Write(OUT y) port



multi-port connector



BIP representation of a system

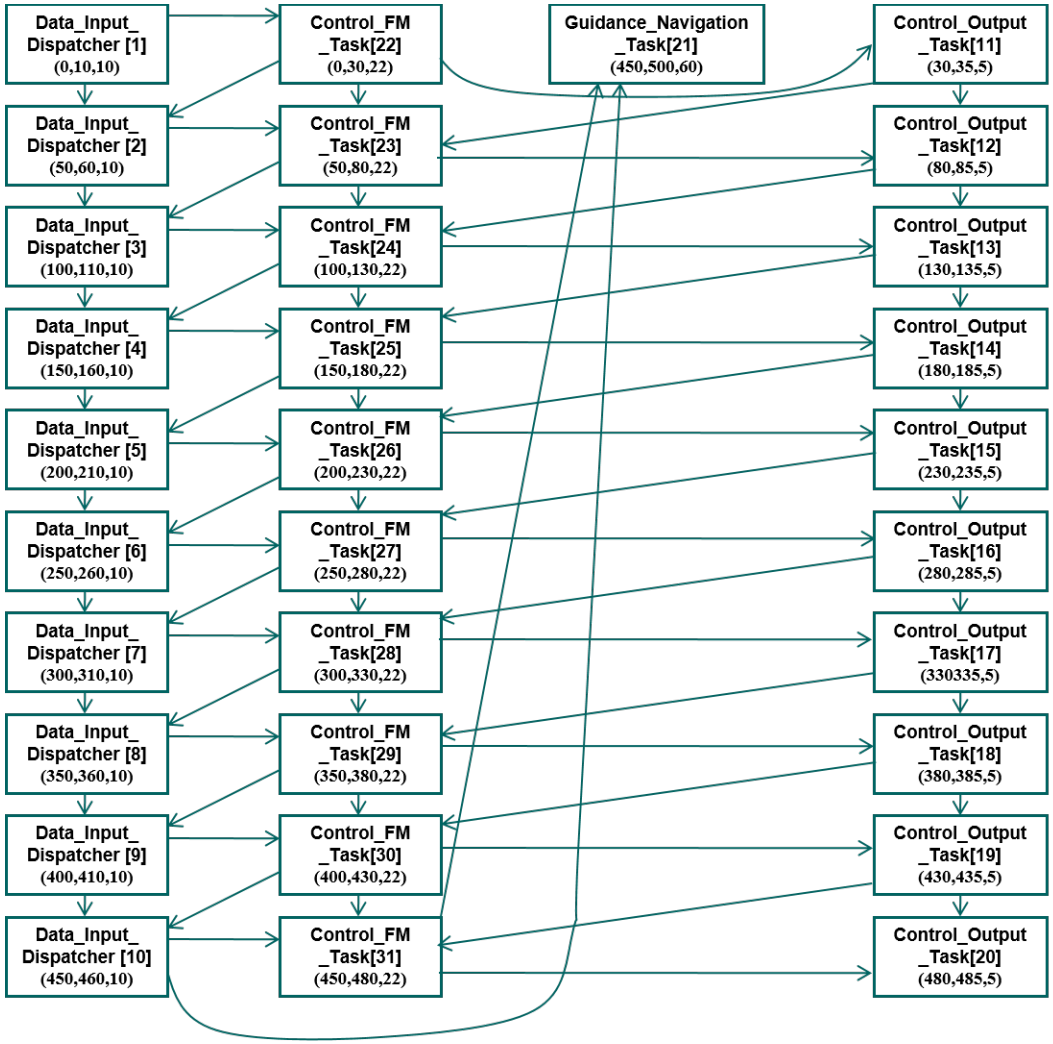


Use case: GNC app by Elecnor Deimos-Space S.L.U

FPPN's Task Graph

computed by the TASTE2DOLC generator

arcs show the access order to interprocess channels by writers and readers



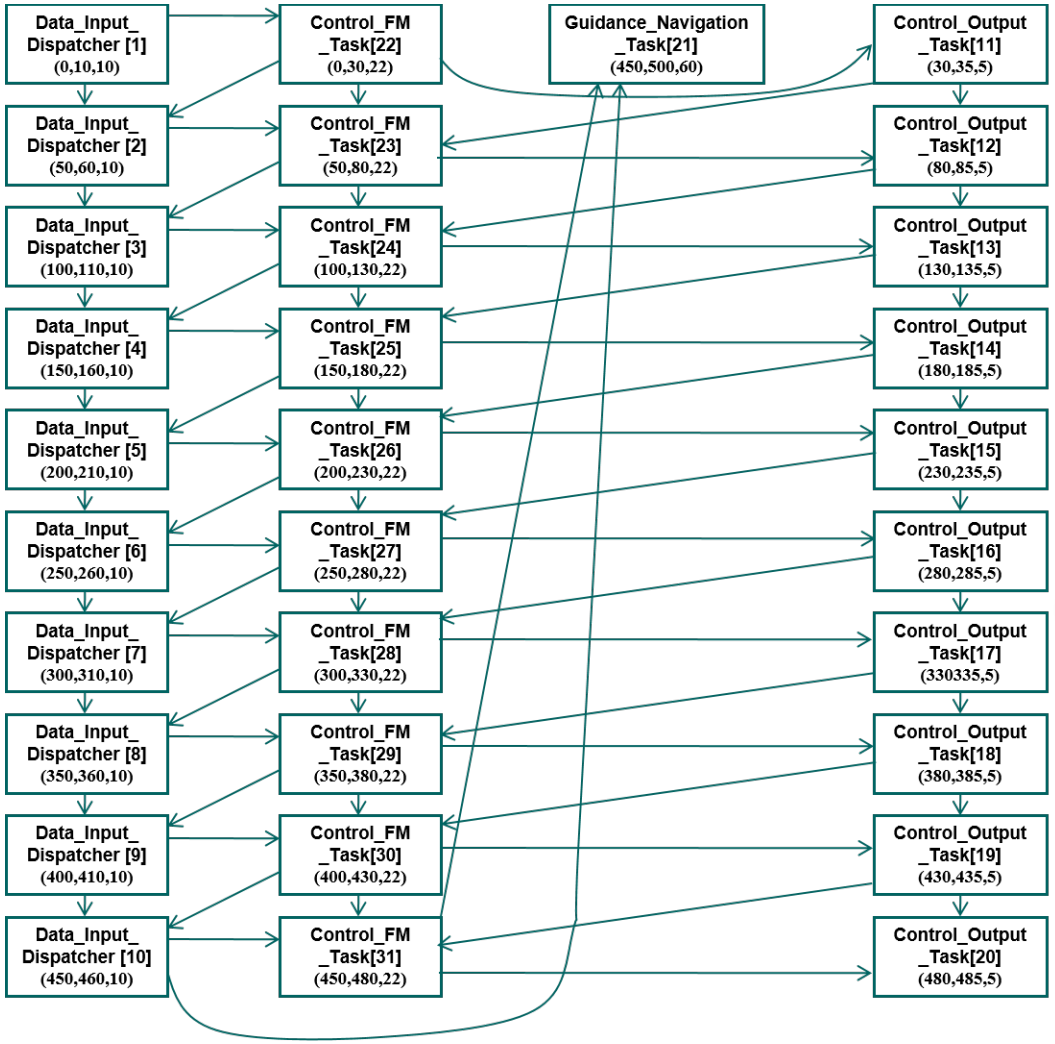
$$p[i] \quad (A_i, D_i, C_i)$$

Use case: GNC app by Elecnor Deimos-Space S.L.U

FPPN's Task Graph

computed by the TASTE2DOLC generator

arcs show the access order to interprocess channels by writers and readers



$p[i]$

(A_i, D_i, C_i)

i-th job of process p

$[A_i, D_i]$ job sched. window

C_i WCET

job's scheduling window is relative to the hyperperiod (LCM of periods 500 ms)

Multi-core interference aspects

- Types of interference (SW and HW resources)
  - coarse-grain – access to shared resource in 'coarse' blocks (once or few times per job execution)

    read data superblock → compute → write data superblock
  - fine-grain – sporadic, can occur many times per job (e.g. bus accesses due to load/store in memory); extra WCET margins
- Overhead of the BIP RTE (coarse-grain, constant block size)

  δ – worst-case time to handle one discrete transition in automaton
- Other interfering resources can be modelled similarly
  - any coarse-grain interference aspect is reflected in our task graph
  - time-triggered scheduling – tasks start at fixed time instants even if previous tasks finish earlier

Multi-core interference aspects

- Types of interference (SW and HW resources)
  - coarse-grain – access to shared resource in 'coarse' blocks (once or few times per job execution)

    read data superblock → compute → write data superblock
  - fine-grain – sporadic, can occur many times per job (e.g. bus accesses due to load/store in memory); extra WCET margins

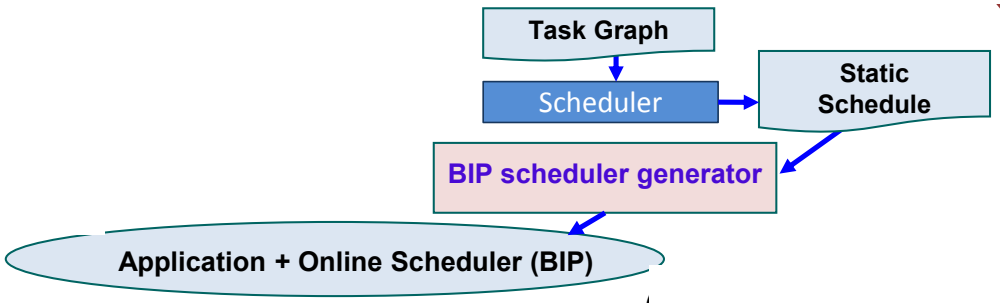
simplifying assumption

- Overhead of the BIP RTE (coarse-grain, constant block size)

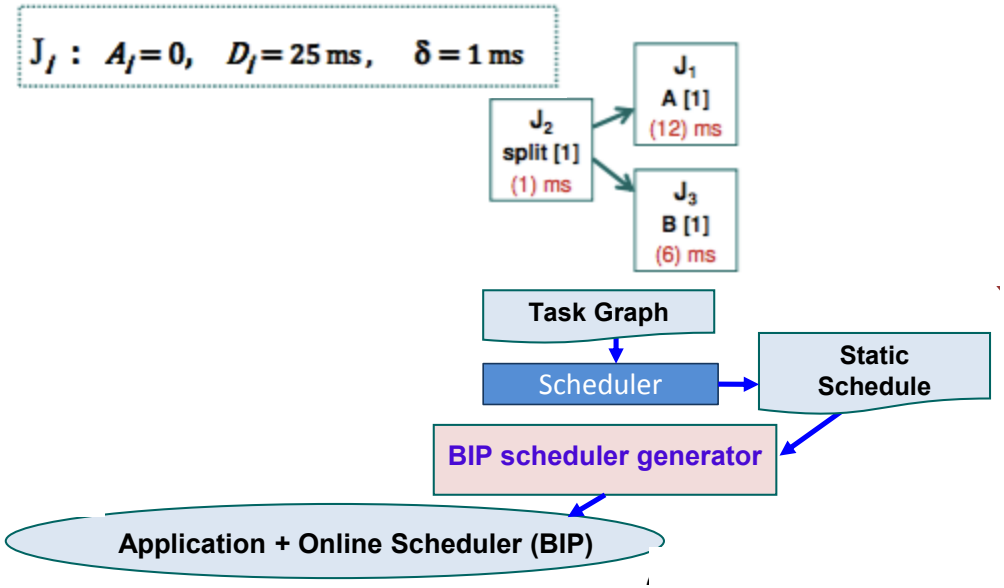
  δ – worst-case time to handle one discrete transition in automaton

- Other interfering resources can be modelled similarly
  - any coarse-grain interference aspect is reflected in our task graph
  - time-triggered scheduling – tasks start at fixed time instants even if previous tasks finish earlier

Schedulability analysis

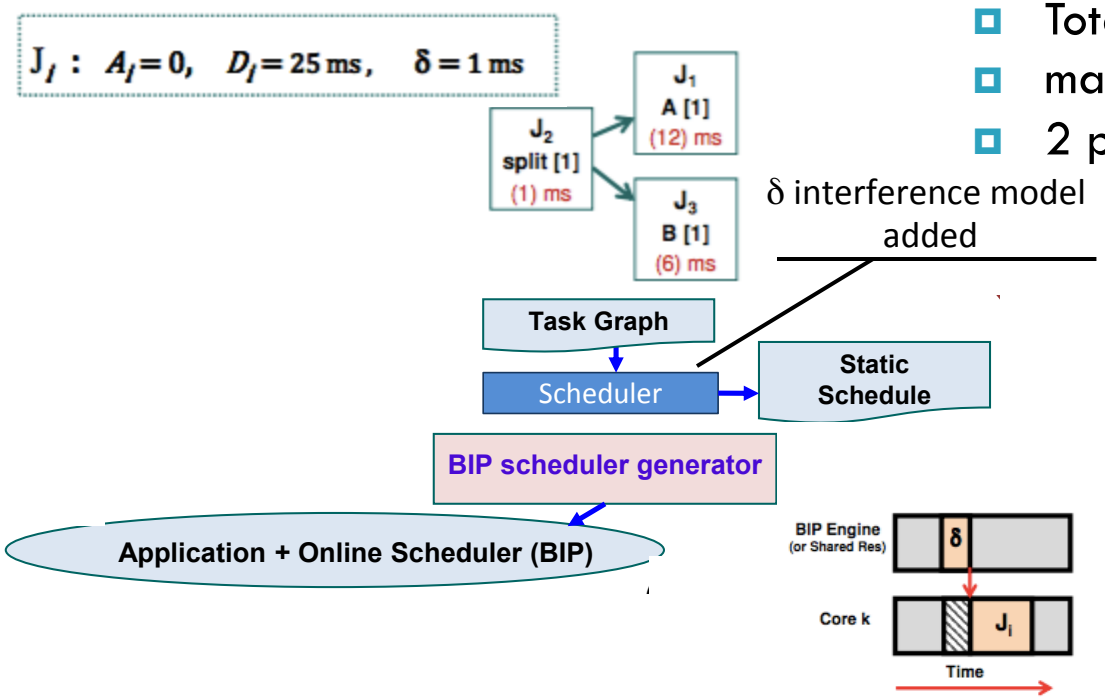


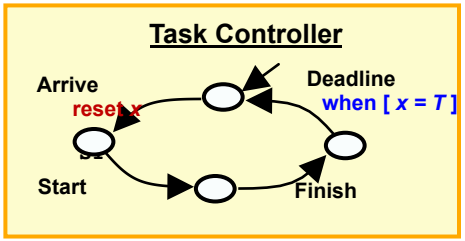
Schedulability analysis

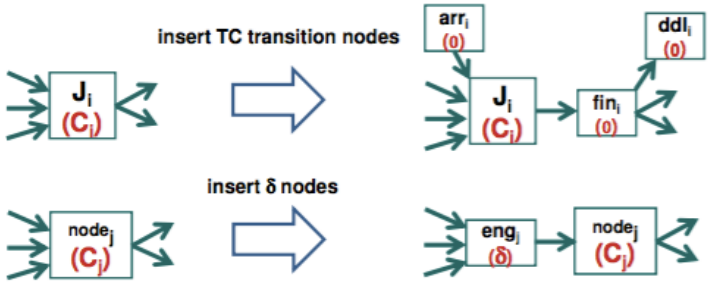


Schedulability analysis

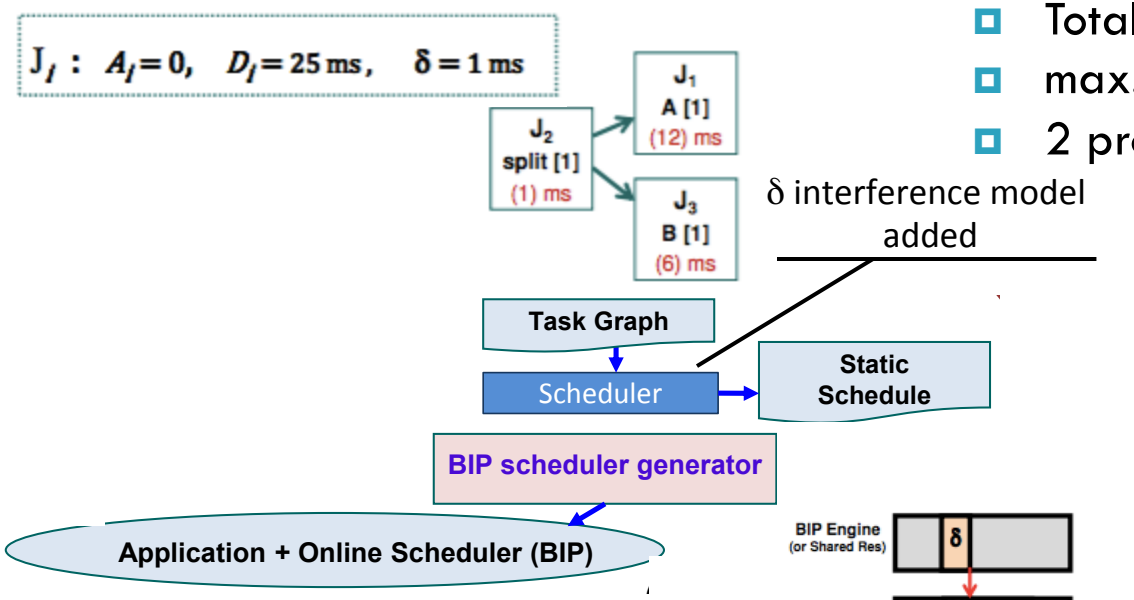
- Total exec. time: 12+1+6+(4·δ)·3 =31 ms
- max. demand-to-capacity=31/25 =124%
- 2 processors needed



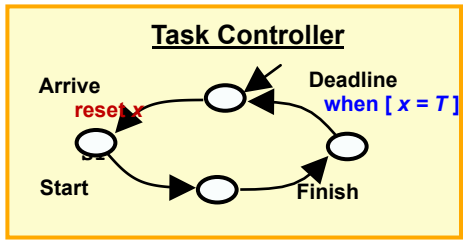


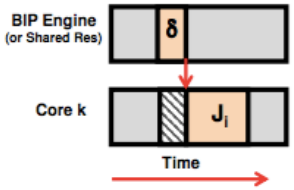


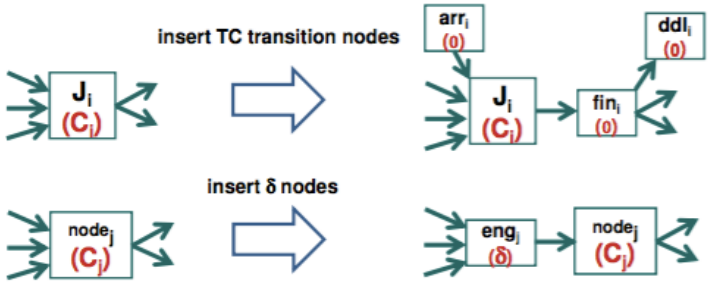
Schedulability analysis



- Total exec. time: 12+1+6+(4·δ)·3 =31 ms
- max. demand-to-capacity=31/25 =124%
- 2 processors needed

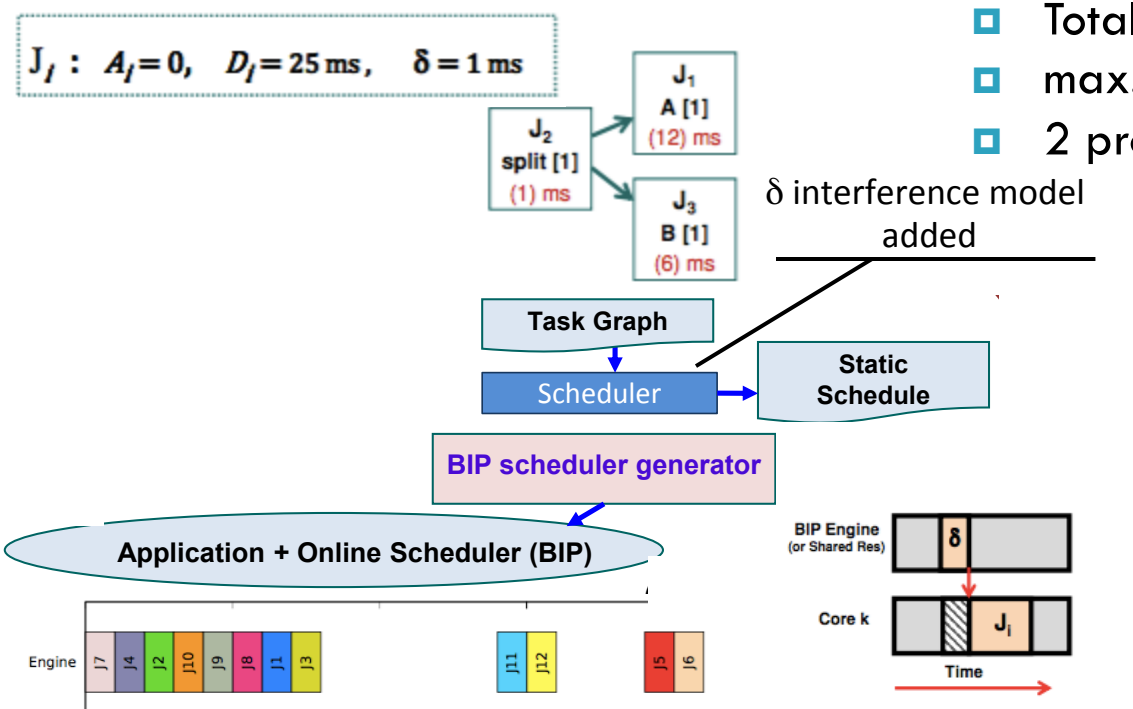




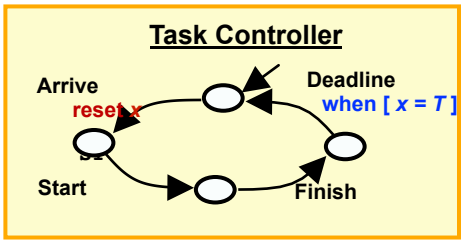


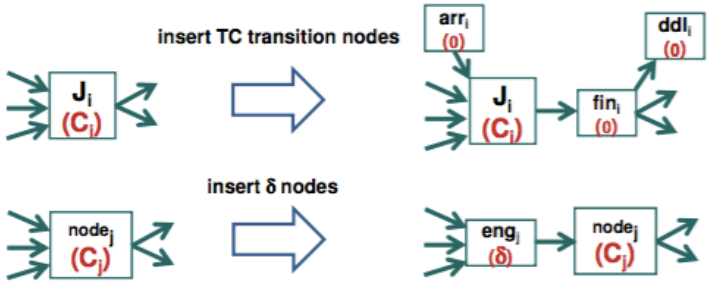
- one control core + pool of compute cores
- List-scheduling (non-preemptive): global fixed-priority simulation with precedence constraints
- Table for time-triggered execution
- Input to the Online Sched. (not yet supported)
- Online Sched. supports resource mngmt policies

Schedulability analysis



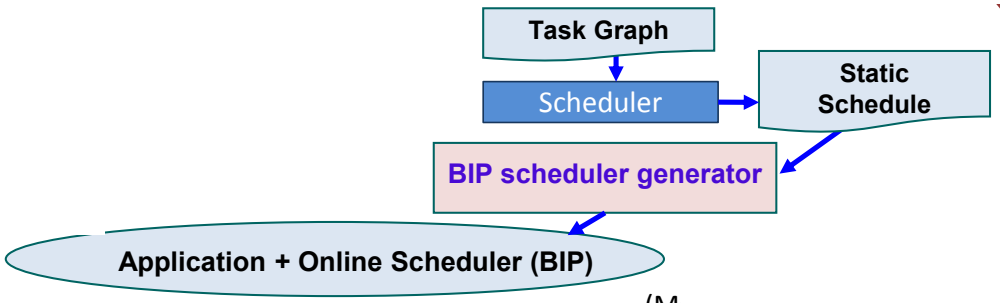
- Total exec. time: 12+1+6+(4·δ)·3 =31 ms
- max. demand-to-capacity=31/25 =124%
- 2 processors needed



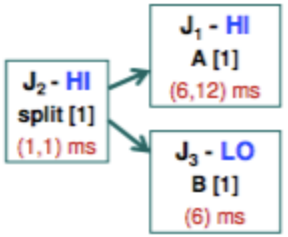


- one control core + pool of compute cores
- List-scheduling (non-preemptive): global fixed-priority simulation with precedence constraints
- Table for time-triggered execution
- Input to the Online Sched. (not yet supported)
- Online Sched. supports resource mngmt policies

Schedulability analysis for mixed-criticality

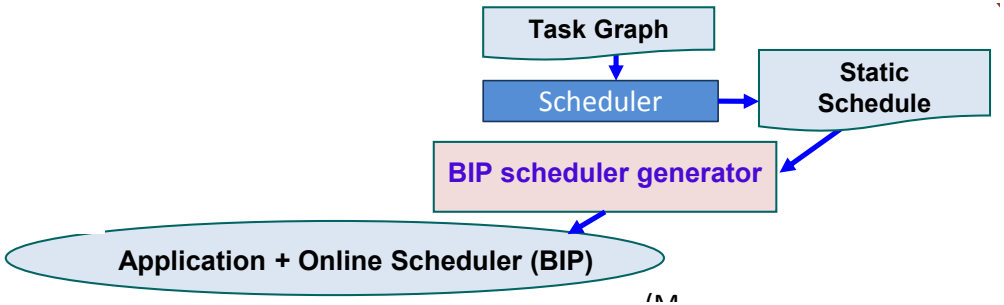


Schedulability analysis for mixed-criticality



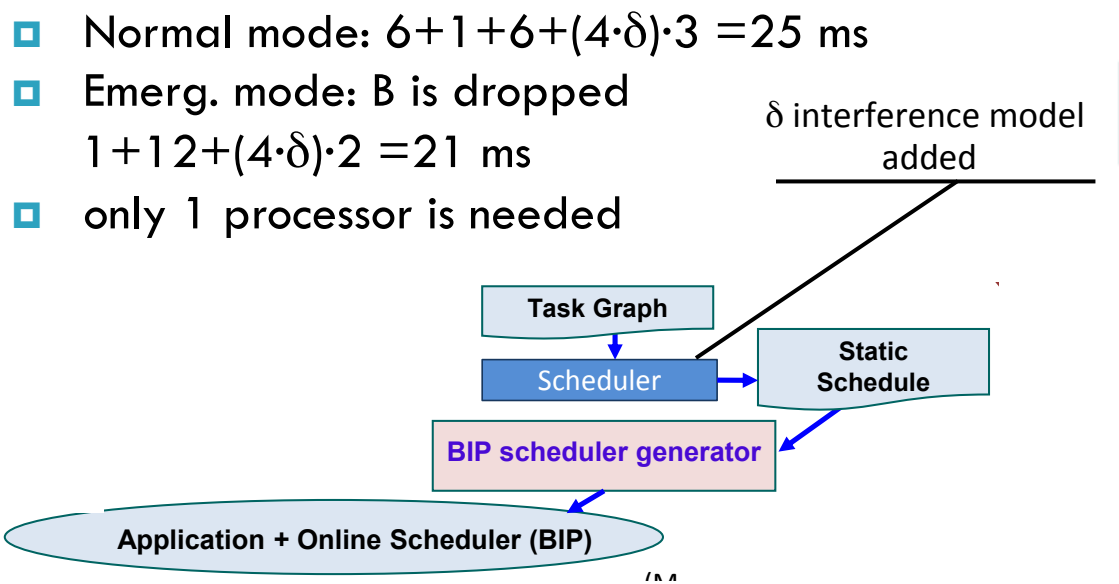
"mixed-critical" variant of task graph

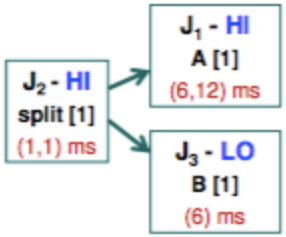
$J_l : A_l = 0, \quad D_l = 25\,\text{ms}, \quad \delta = 1\,\text{ms}$



Schedulability analysis for mixed-criticality

- Normal mode: $6+1+6+(4\cdot\delta)\cdot3 = 25$ ms
- Emerg. mode: B is dropped
  $1+12+(4\cdot\delta)\cdot2 = 21$ ms
- only 1 processor is needed



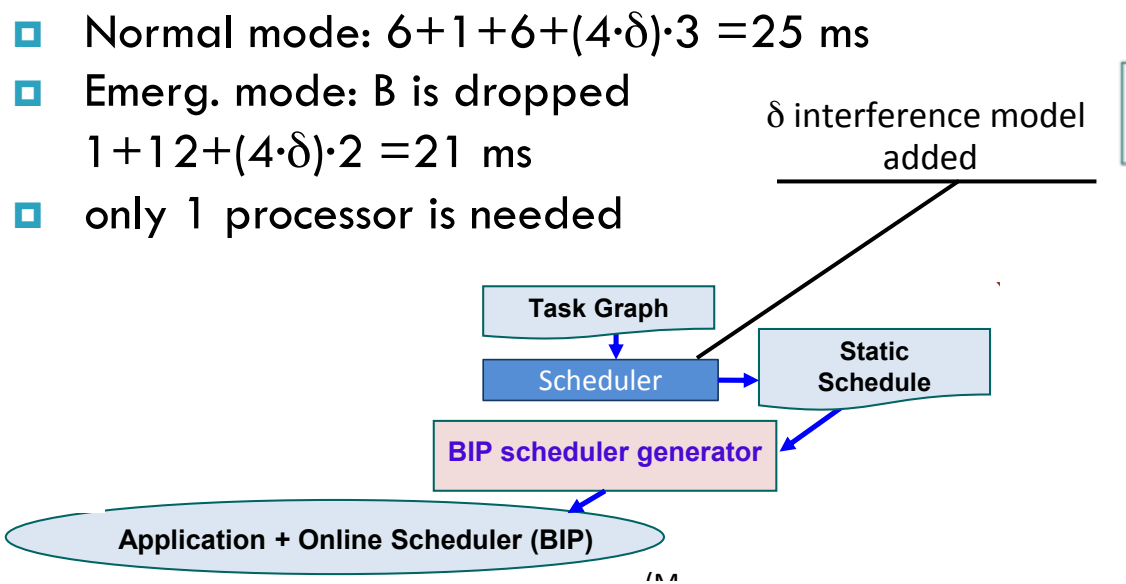


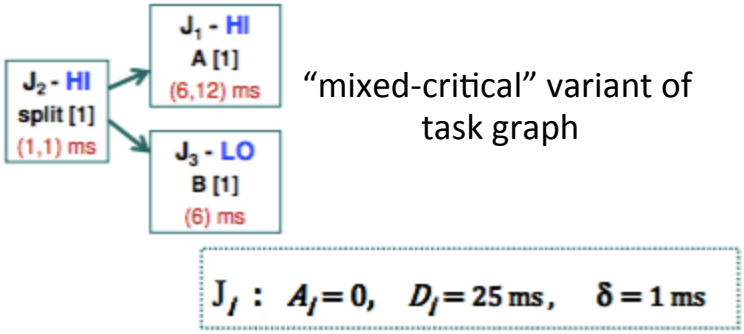
"mixed-critical" variant of task graph

$J_l : A_l = 0, \quad D_l = 25\,\text{ms}, \quad \delta = 1\,\text{ms}$

Schedulability analysis for mixed-criticality

- Normal mode: 6+1+6+(4·δ)·3 =25 ms
- Emerg. mode: B is dropped
  1+12+(4·δ)·2 =21 ms
- only 1 processor is needed



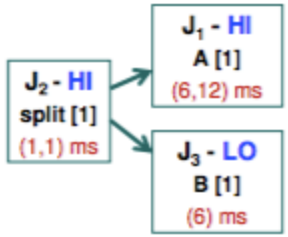


- Normal-mode table is first generated
- Emergency-mode table:
  mode switch can happen, while HI jobs continue without preemption or migration to other core
- Only HI-to-HI precedencies taken into account
- Same job-to-core mapping as in normal mode
- Schedulability fails upon detecting deadline miss

Schedulability analysis for mixed-criticality

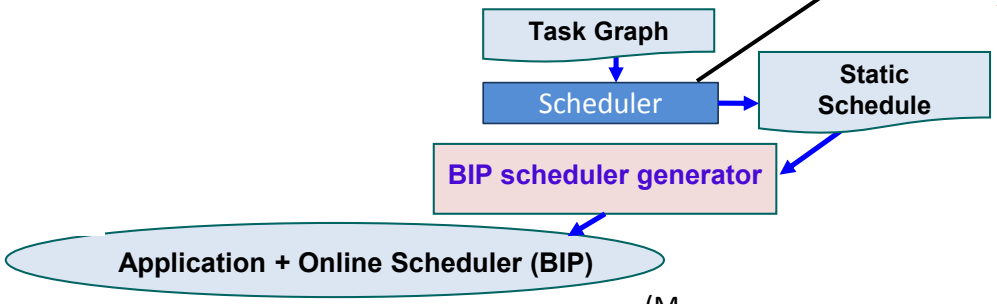
- Normal mode: 6+1+6+(4·δ)·3 =25 ms
- Emerg. mode: B is dropped
  1+12+(4·δ)·2 =21 ms
- only 1 processor is needed

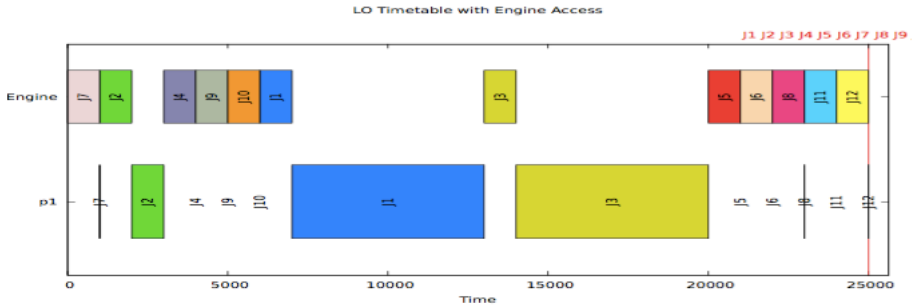
δ interference model added



"mixed-critical" variant of task graph

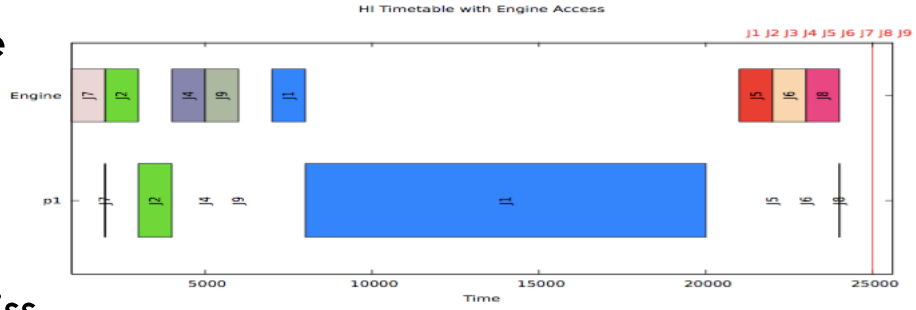
$J_l: A_l = 0, \quad D_l = 25\,\text{ms}, \quad \delta = 1\,\text{ms}$





(b) MC Schedule: Normal Mode

- Normal-mode table is first generated
- Emergency-mode table:
  mode switch can happen, while HI jobs continue without preemption or migration to other core
- Only HI-to-HI precedencies taken into account
- Same job-to-core mapping as in normal mode
- Schedulability fails upon detecting deadline miss



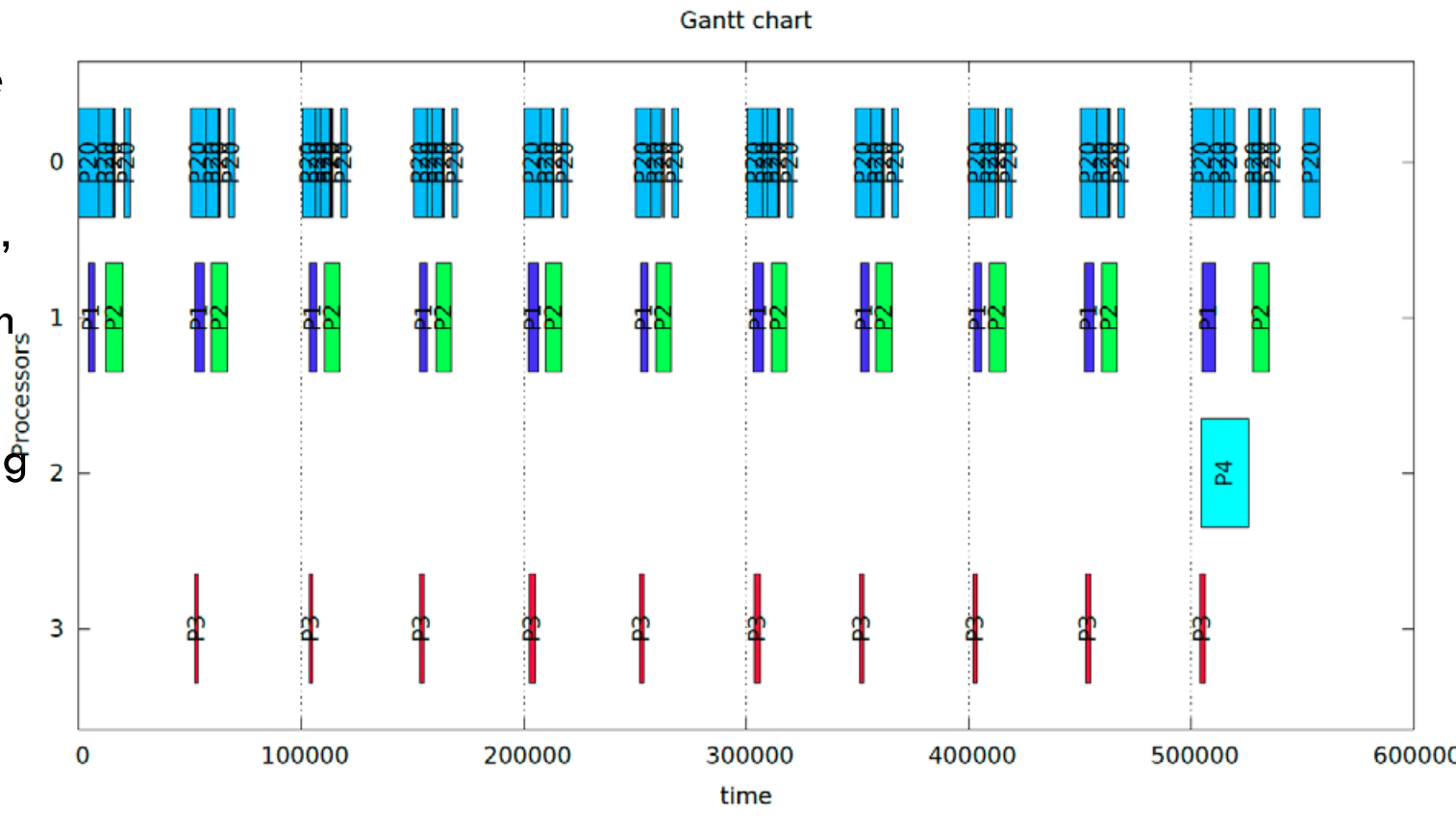
(c) MC Schedule: Emergency Mode

Use case: GNC app by Elecnor Deimos-Space S.L.U

Real-time execution on the Leon4

"pipelined version"

Modified execution order without violating the data dependency among tasks.



P1 – Data Input Dispatcher

P2 – Control FM

P3 – Control Output

P4 – Guidance Navigation

core-0: BIT RTE

Use case: GNC app by Elecnor Deimos-Space S.L.U

Real-time execution on the Leon4

"pipelined version"

Modified execution order without violating the data dependency among tasks.

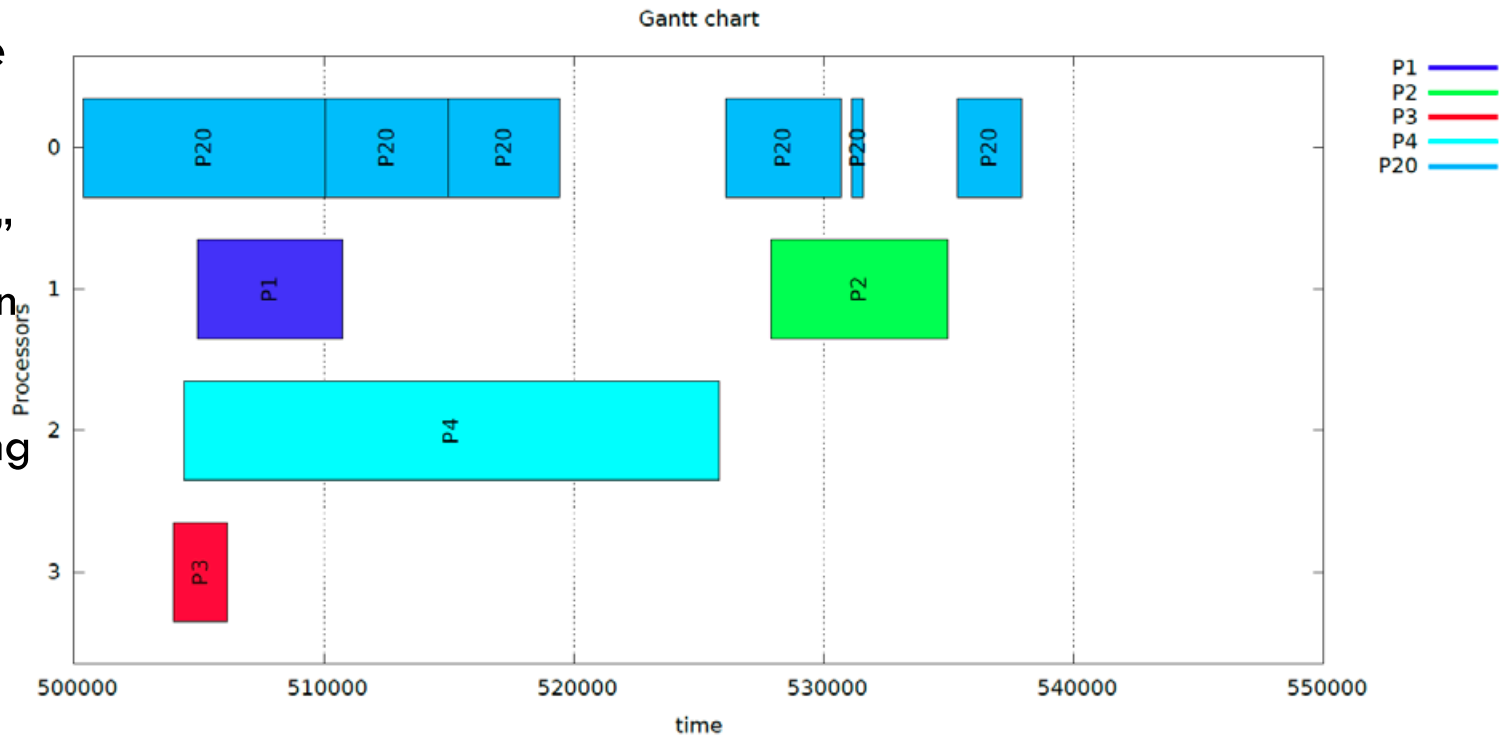
P1 – Data Input Dispatcher

P2 – Control FM

P3 – Control Output

P4 – Guidance Navigation

core-0: BIT RTE



Zoom-in the time frame with parallel execution

Conclusions

- MoSATT-CMP design flow:
  - design-by-refinement of multicore systems
  - TASTE tool-chain & executable formal models in BIP
  - Schedulability analysis that takes into account
    - diverse task dependency patterns (MoCs)
    - sources of coarse-grain & fine-grain interference
    - mixed-criticality aspects
  - Code generation for BIP RTE: what you verify is what you execute
  - New measurement-based WCET analysis guaranteeing probabilistic estimates
  - New cache interference analysis technique for the Leon4
  - Validation on an industrial GNC application running on the Leon4

Future work

- Improve integration of the FPPN and other MoCs with the TASTE Interface View modelling semantics.
- Integration with the Ocarina TASTE model processing library.
- Explore further the potentials of execution time analysis.
- Support for preemptive scheduling by BIP/BIP RTE.
- Multicore schedulability of communicating system nodes.
- Integration of BIP RTE with the TASTE PolyORB-HI middleware to handle communication between distributed multicore nodes.
- Improve the tool support to achieve a higher TRL.

Centre For Research & Technology Hellas (CERTH)

- Panagiotis Katsaros, Prof.
- Lefteris Angelis, Prof.
- Spyros Nikolaidis, Prof.
- Alexandros Zerzelidis, Researcher
- Fotis Gioulekas, Researcher
- Maria Ntogramatzi, PhD student

Verimag Laboratory Université Grenoble-Alpes - France

- Saddek Bensalem, Prof.
- Peter Poplavko, Researcher
- Marius Bozga, Researcher

Elecnor Deimos-Space S.L.U, Spain

- Pedro Palomo, Software Engineer

Cobham Gaisler AB, Sweden

- Jan Andersson, Engineer
- Daniel Hellstrom, Software Engineer

Contact: katsaros@csd.auth.gr

https://www.researchgate.net/project/Task-Automata-BIP-Co-programming-of-Applications-and-Online-Schedulers-in-Timing-Critical-Systems

Schedulability Analysis Techniques & Tools for Cached and Multicore Processors

ESA/ESTEC - Dec. 6, 2016