Virtual Execution of AADL Models via a Translation into Synchronous Programs

E. Jahier, N. Halbwachs, P. Raymond, X. Nicollin
Verimag, Grenoble

D. Lesens
Astrium Space Transportation, Les Mureaux

Synchrony and Asynchrony (1/3)

Synchronous languages and associated tools (Scade, Esterel-Studio, Sildex, . . .) are well-established for centralized, statically scheduled applications

What about more complex situations?

- Need for dynamic scheduling: urgent sporadic events, multiple periods
- Need for distribution: redundancy, performances, physical constraints

Synchrony and Asynchrony (2/3)

In real-time systems, purely asynchronous situations are rare

Partial synchrony, or strongly constrained asynchrony: e.g.,

- known periods
- known clock drift
- quite precise WCET

Synchrony and Asynchrony (3/3)

Related works:

Synchrony and Asynchrony (3/3)

Related works:

- extend the synchronous model
  CRP [Berry-Shyamasundar-Ramesh],
  Multiclock-Esterel [Berry-Sentovitch],
  n-synchrony [Cohen-Duranton-Eisenbeis-Pagetti-Plateau-Pouzet],
  GALS [Metropolis], [Polychrony],
  Tag machines [Benveniste-Caillaud-Carloni-Sangiovanni]

Synchrony and Asynchrony (3/3)

Related works:

- extend the synchronous model
  CRP [Berry-Shyamasundar-Ramesh],
  Multiclock-Esterel [Berry-Sentovitch],
  n-synchrony [Cohen-Duranton-Eisenbeis-Pagetti-Plateau-Pouzet],
  GALS [Metropolis], [Polychrony],
  Tag machines [Benveniste-Caillaud-Carloni-Sangiovanni]
- less synchronous implementations
  Multi-task implementations [SYNDEX], [Caspi-Scaife],
  Distributed code [Caspi-Girault],[Caspi-Salem], [Potop-Caillaud]

Synchrony and Asynchrony (3/3)

Related works:

- extend the synchronous model
  CRP [Berry-Shyamasundar-Ramesh],
  Multiclock-Esterel [Berry-Sentovitch],
  n-synchrony [Cohen-Duranton-Eisenbeis-Pagetti-Plateau-Pouzet],
  GALS [Metropolis], [Polychrony],
  Tag machines [Benveniste-Caillaud-Carloni-Sangiovanni]
- less synchronous implementations
  Multi-task implementations [SYNDEX], [Caspi-Scaife],
  Distributed code [Caspi-Girault],[Caspi-Salem], [Potop-Caillaud]
- model asynchrony within the synchronous framework
  SafeAir, SafeAir-II projects [Baufreton et-al],
  Polychrony [Le Guernic-Talpin-Le Lann], [Gamatié-Gautier],
  this talk (same approach, in the ctxt of the Assert project)

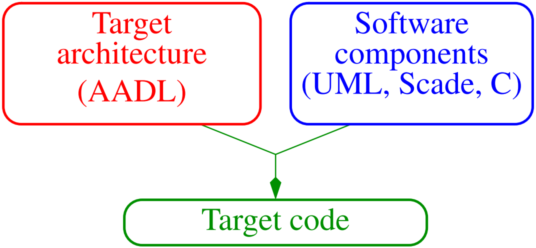
The ASSERT Project (1/2)

- European “Integrated Project” on model-driven design of embbedded systems
- Main application domain: aerospace applications

The ASSERT Project (1/2)

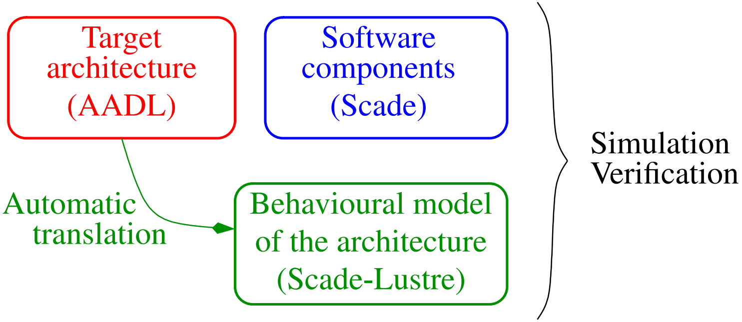
European "Integrated Project" on model-driven design of embbedded systems

Main application domain: aerospace applications



The ASSERT Project (2/2)

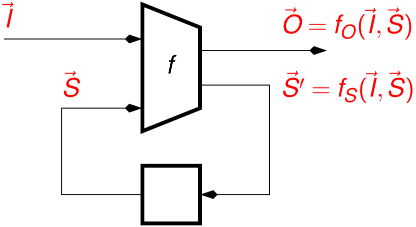
What this talk is about:



The Synchronous Paradigm (1/2)

Synchronous machines

Basic components: generalized Mealy machines



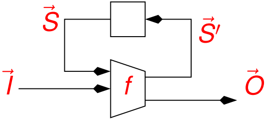
Behaviour: $(\vec{S}_0, \vec{I}_0, \vec{O}_0), (\vec{S}_1, \vec{I}_1, \vec{O}_1), \dots, (\vec{S}_n, \vec{I}_n, \vec{O}_n), \dots,$

with $O_n = f_O(\vec{I}_n, \vec{S}_n)$ and $\vec{S}_{n+1} = f_S(\vec{I}_n, \vec{S}_n)$

The Synchronous Paradigm (2/2)

Synchronous machines

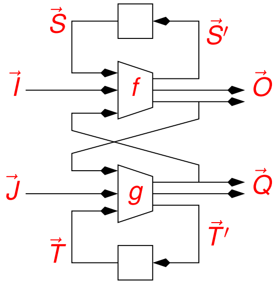
Parallel composition:



The Synchronous Paradigm (2/2)

Synchronous machines

Parallel composition:



$$(\vec{S}', \vec{O}) = f(\vec{I}, \vec{S}, \vec{Q})$$
$$(\vec{T}', \vec{Q}) = g(\vec{J}, \vec{T}, \vec{O})$$

(deterministic, provided there is no combinational loop)

Synchronous Modelling of Asynchrony (1/4)

Need to

- prevent a component from reacting (sporadic reactions)
- non-determinism
- model execution time

Synchronous Modelling of Asynchrony (2/4)

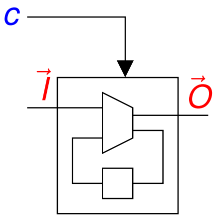
Prevent a component from reacting

- available in all synchronous languages:
  - clocks in Lustre and Signal
  - activation conditions in Scade
  - suspend statement in Esterel

Synchronous Modelling of Asynchrony (3/4)

Activation condition in Scade

A distinguished Boolean input, say c, decides if the component must react.

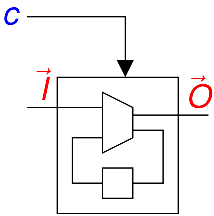


Synchronous Modelling of Asynchrony (3/4)

Activation condition in Scade

A distinguished Boolean input, say c, decides if the component must react.

- when $c = 1$ the normal reaction occurs

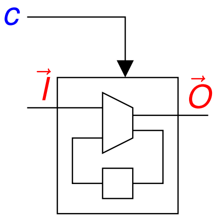


Synchronous Modelling of Asynchrony (3/4)

Activation condition in Scade

A distinguished Boolean input, say c, decides if the component must react.

- when $c = 1$ the normal reaction occurs
- when $c = 0$
  - the state does not change

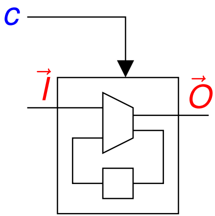


Synchronous Modelling of Asynchrony (3/4)

Activation condition in Scade

A distinguished Boolean input, say c, decides if the component must react.

- when $c = 1$ the normal reaction occurs
- when $c = 0$
  - the state does not change
  - the output keeps its previous value

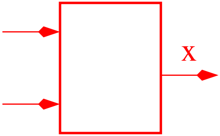


Synchronous Modelling of Asynchrony(4/4)

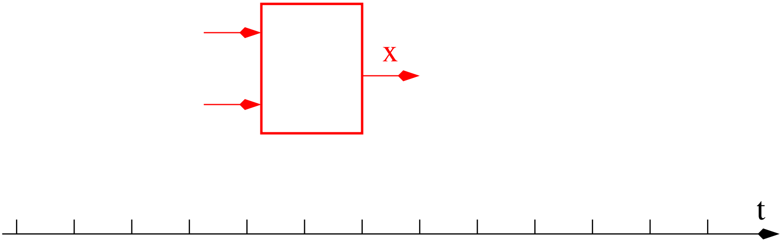
Non determinism

- Just by adding auxiliary inputs (oracles)
- Restriction of non-determinism:
  - constraints/assumptions on oracles ensured by “assertions” or transducer (scheduler)

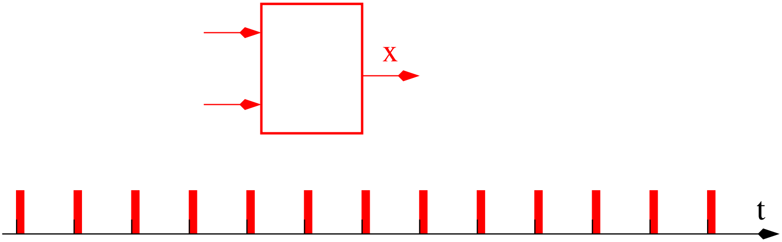
A task in the synchronous world



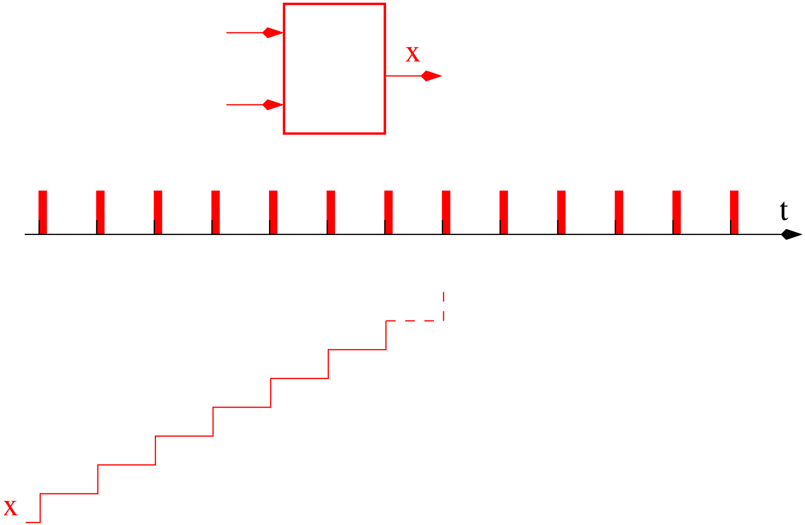
A task in the synchronous world



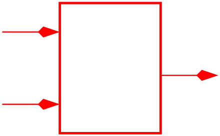
A task in the synchronous world



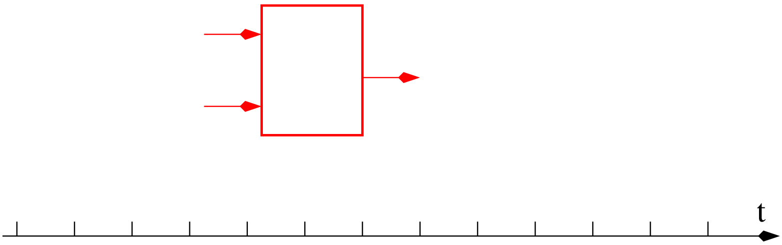
A task in the synchronous world



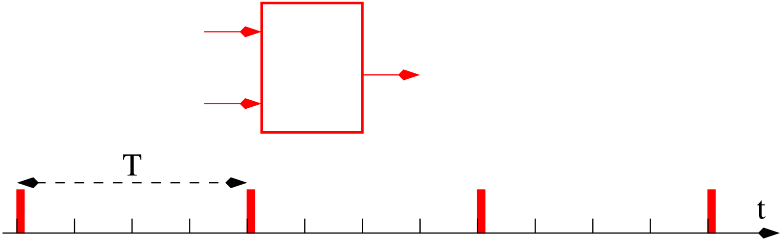
A sporadic or periodic task



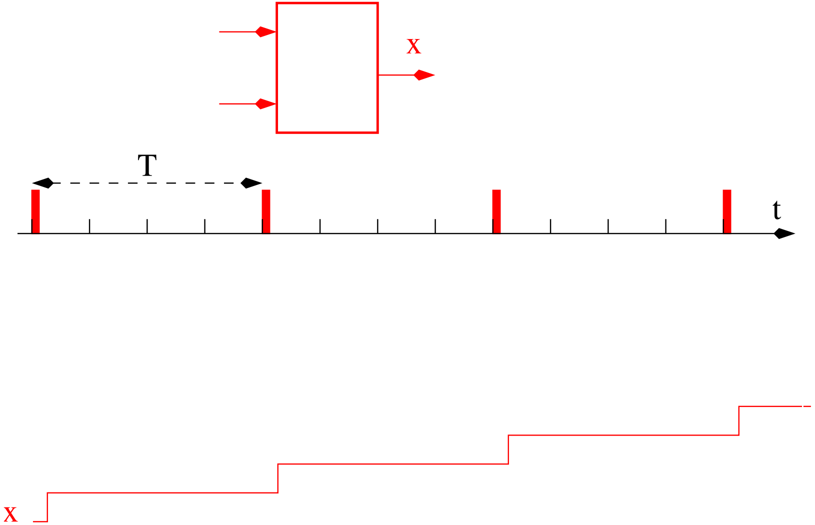
A sporadic or periodic task



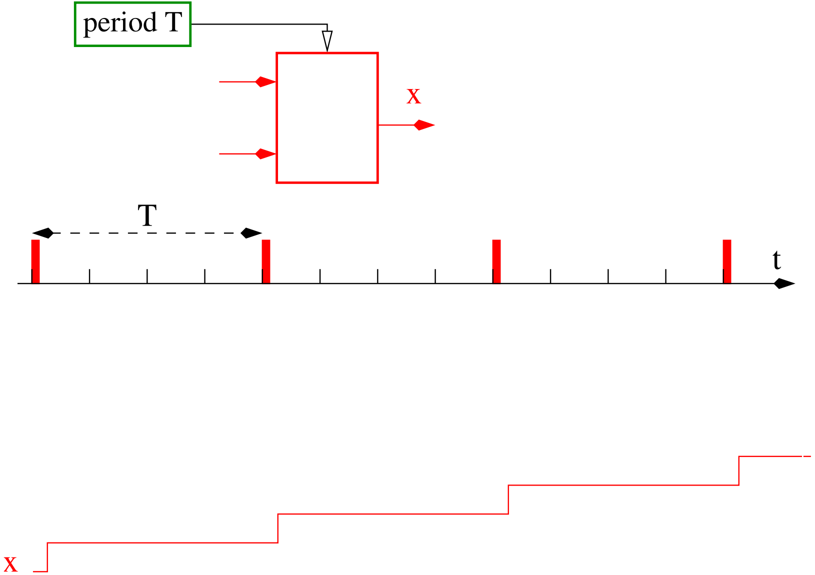
A sporadic or periodic task



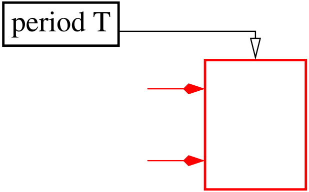
A sporadic or periodic task



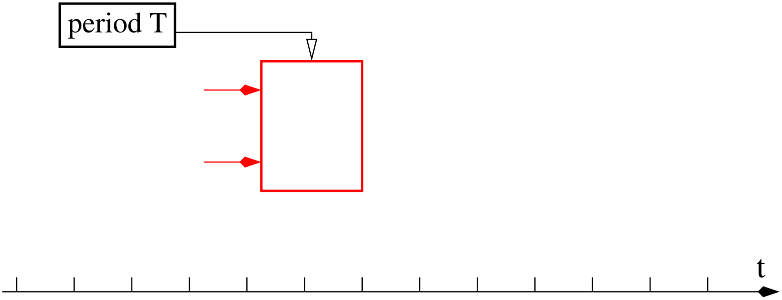
A sporadic or periodic task



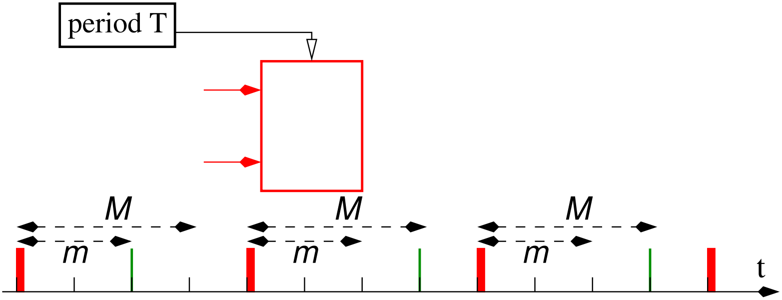
Execution time



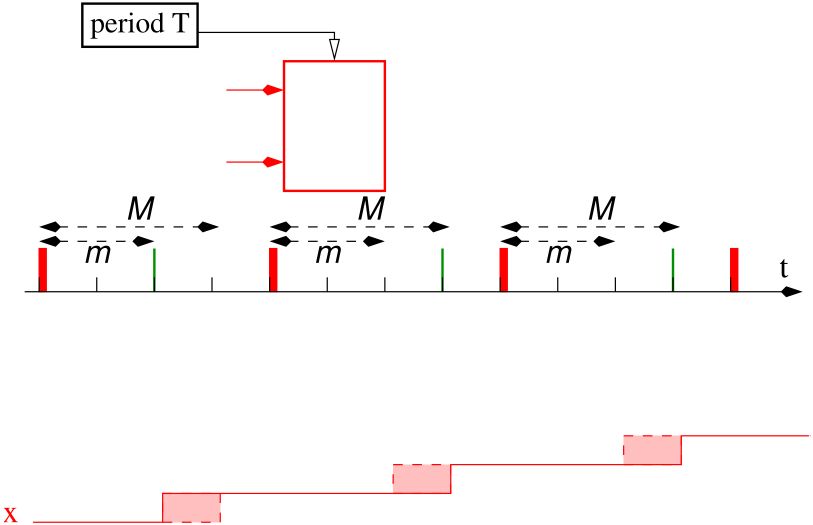
Execution time



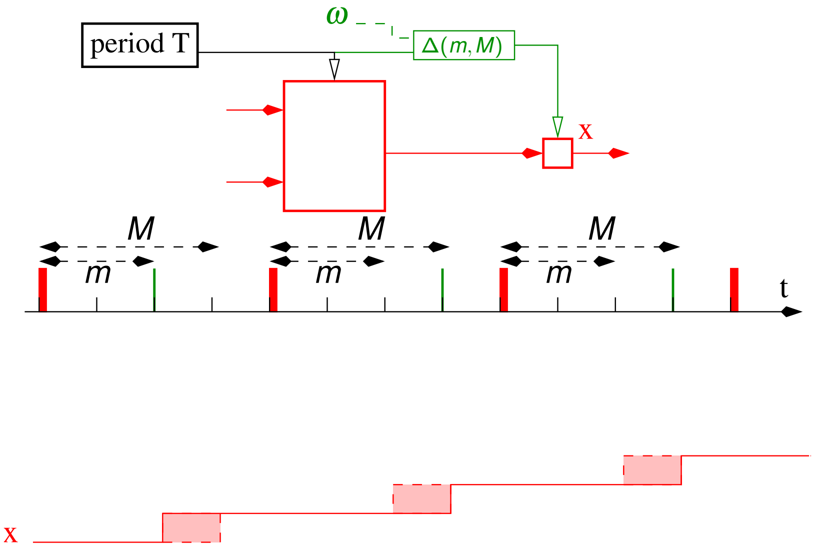
Execution time



Execution time

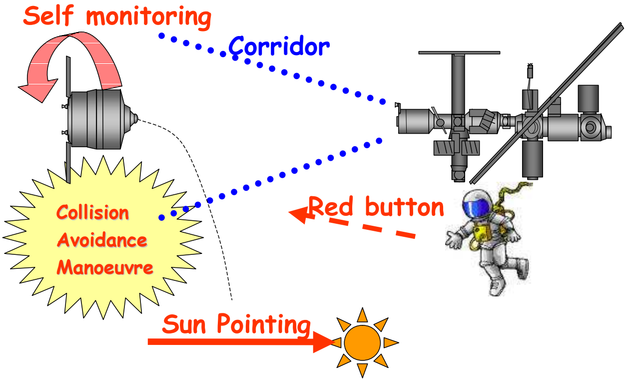


Execution time



The PFS case study (1/4)

- Proximity Flight Safety (PFS), part of the Automatic Transfer Vehicule (ATV), spacecraft in charge of supplying the International Space Station (ISS) ESA, Astrium-ST
- Ensures the safety of the approach of the ATV to the ISS (most safety critical part of the mission)



When anything goes wrong, the PFS is in charge of safely moving the ATV apart from the ISS, and to orient it towards the sun ("Collision Avoidance Manoeuvre", CAM)

The PFS case study (3/4)

The system is made of two redundant "Monitoring and Safety Units" (MSU): one master, one backup

Each MSU:

- detects anomalies: failures of the main computer, abnormal state of the bus, erroneous position or speed of the ATV, "red button" pressed from inside the ISS
- detects its own failures (master change)
- is able to perform a CAM

The PFS case study (4/4)

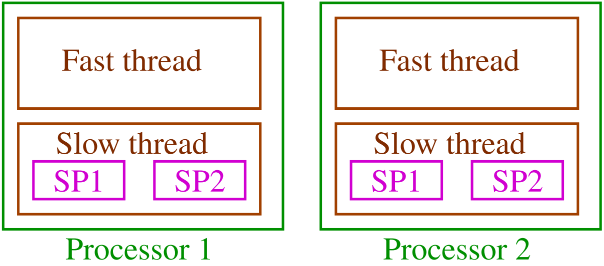
Distribution: Two computers (one for each MSU) running in quasi-synchrony

Multitasking: Each MSU consists of two periodic tasks (one fast, one slow). Each task specified in Scade

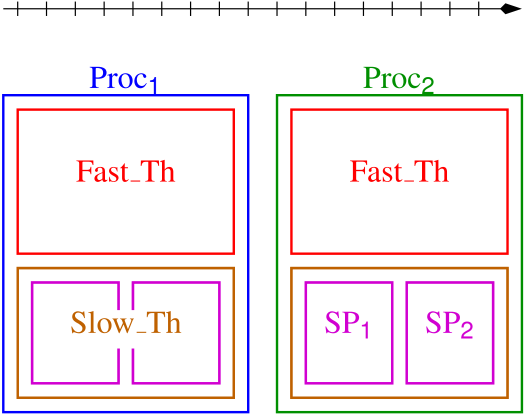
The PFS case study (4/4)

Distribution: Two computers (one for each MSU) running in quasi-synchrony

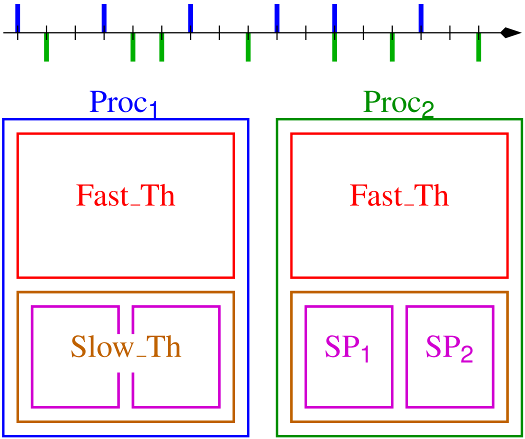
Multitasking: Each MSU consists of two periodic tasks (one fast, one slow). Each task specified in Scade



Processes: actual clocks

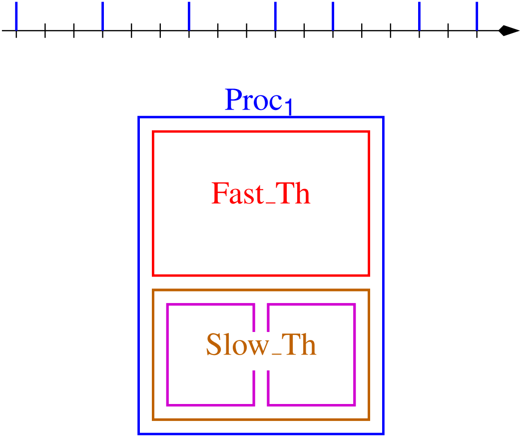


Processes: actual clocks

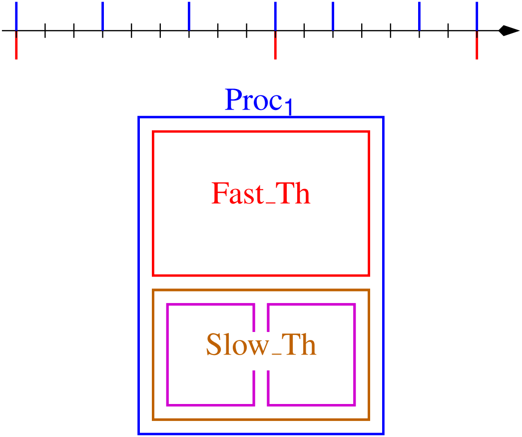


"Quasi-synchronous" clocks
used to count periods and deadlines

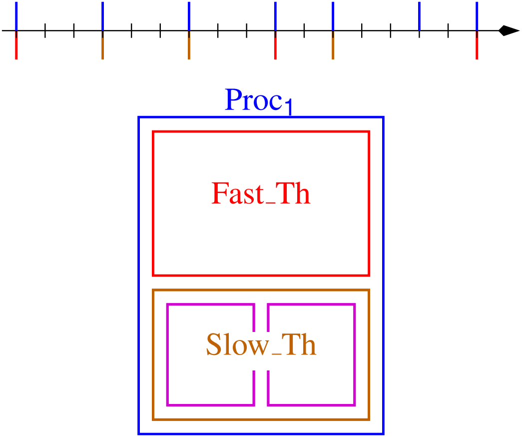
Threads: sharing the processor



Threads: sharing the processor

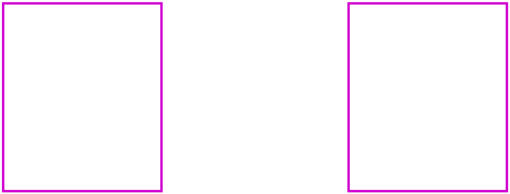


Threads: sharing the processor

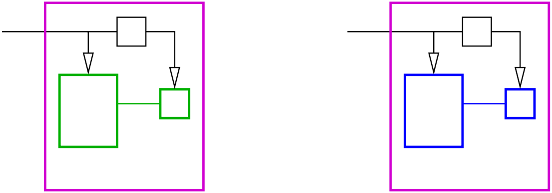


Activity clocks, used to count execution times

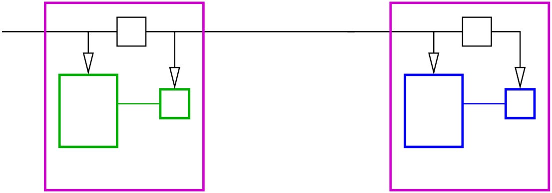
Subprograms: sequencing



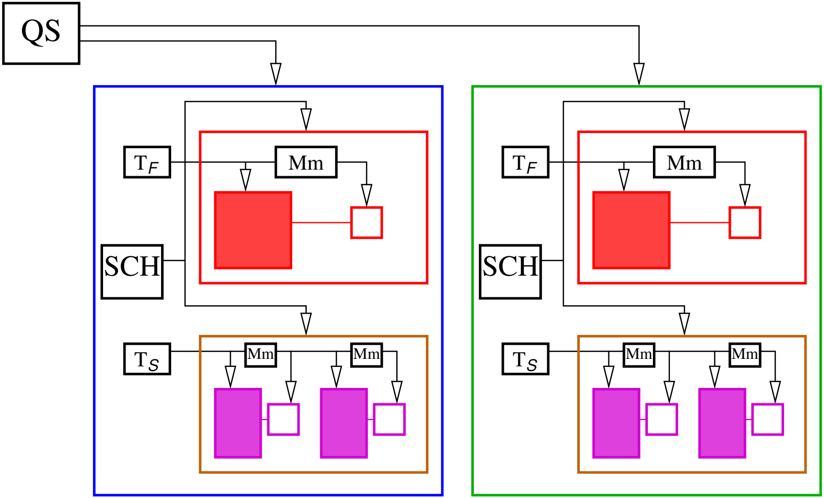
Subprograms: sequencing



Subprograms: sequencing



Final model



Applications

- extensive simulation
  (using the tool LURETTE to generate oracles automatically)
- automatic verification
  - Example of property of the PFS:
    "at each instant, one and only one MSU is the master"

Applications

- extensive simulation
  (using the tool LURETTE to generate oracles automatically)
- automatic verification
  - Example of property of the PFS:
    "at each instant, one and only one MSU is the master"
    Wrong, because of asynchrony.
    Right property:
    "at each instant, there is at most one master"
    "there are at most two clock cycles without master"

Current work

- deterministic communication
- resource management
- scheduling policies

Current work

- deterministic communication
- resource management
- scheduling policies

Conclusion

- Gives precise semantics to AADL
- Makes it executable (early simulation/validation)
- One more non-synchronous application of synchrony