

CoCoSim, a code generation framework for control/command applications

An overview of CoCoSim for multi-periodic discrete Simulink models



Hamza Bourbough, Pierre-Loïc Garoche,
Thomas Loquen, Eric Noulard and Claire
Pagetti

January 31st 2020

ERTS 2020



Outline

☐ Introduction

- **Context**
- Contribution

☐ Reminder on Simulink and synchronous languages

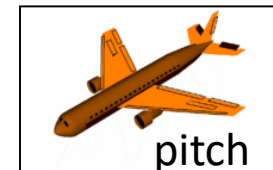
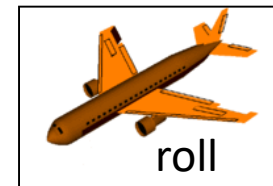
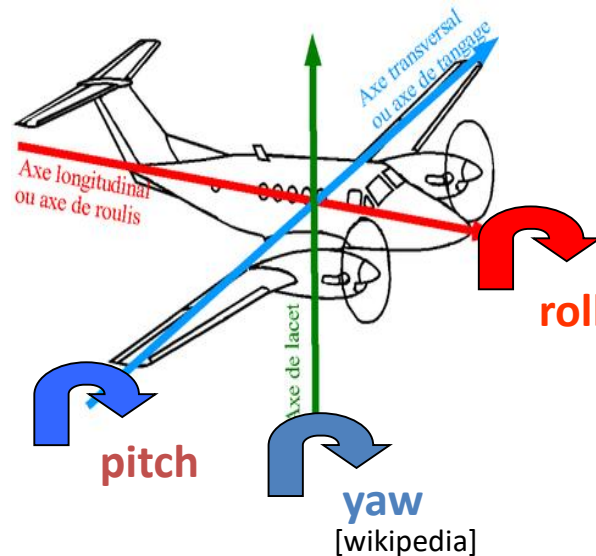
☐ CoCoSim for multi-periodic systems

☐ Two open source use cases: ROSACE and Space shuttle AOCS

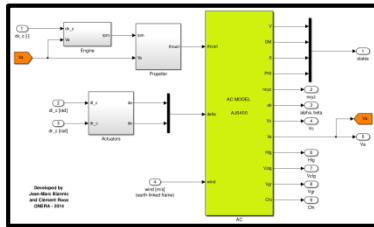
☐ Conclusion

Context – control/command applications

- Control / command applications
 - Safety-critical with DAL – Design Assurance Level A
 - Under certification, and certification development process
- Example: flight control system



Current development cycle

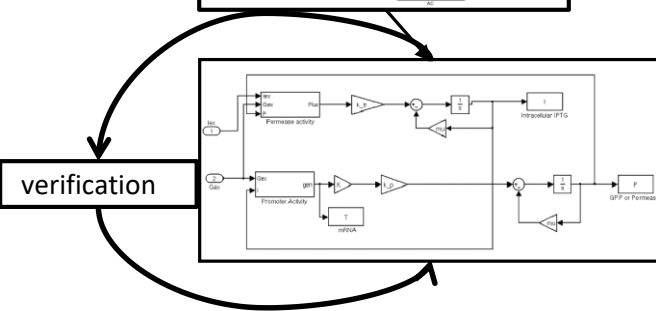


High-level design – control engineering

Implementation

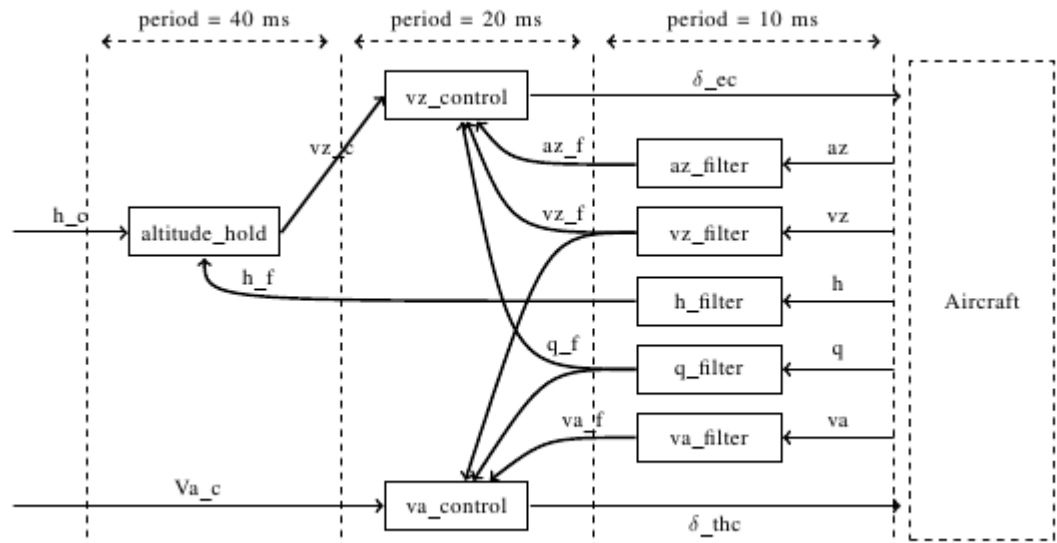
Steps:

- Coding: elementary blocks with Lustre/Scade and multi-periodic assemblies with ad hoc language
- Verification

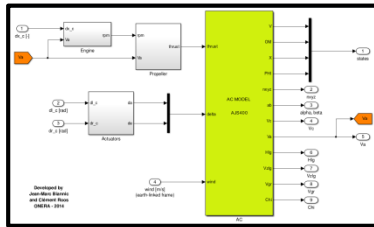


Example: flight control systems

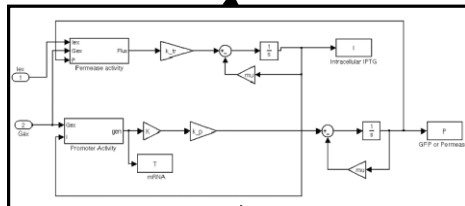
multi-periodic, large size, under temporal and precedence constraints.



Current development cycle



High-level design – control engineering



Implementation



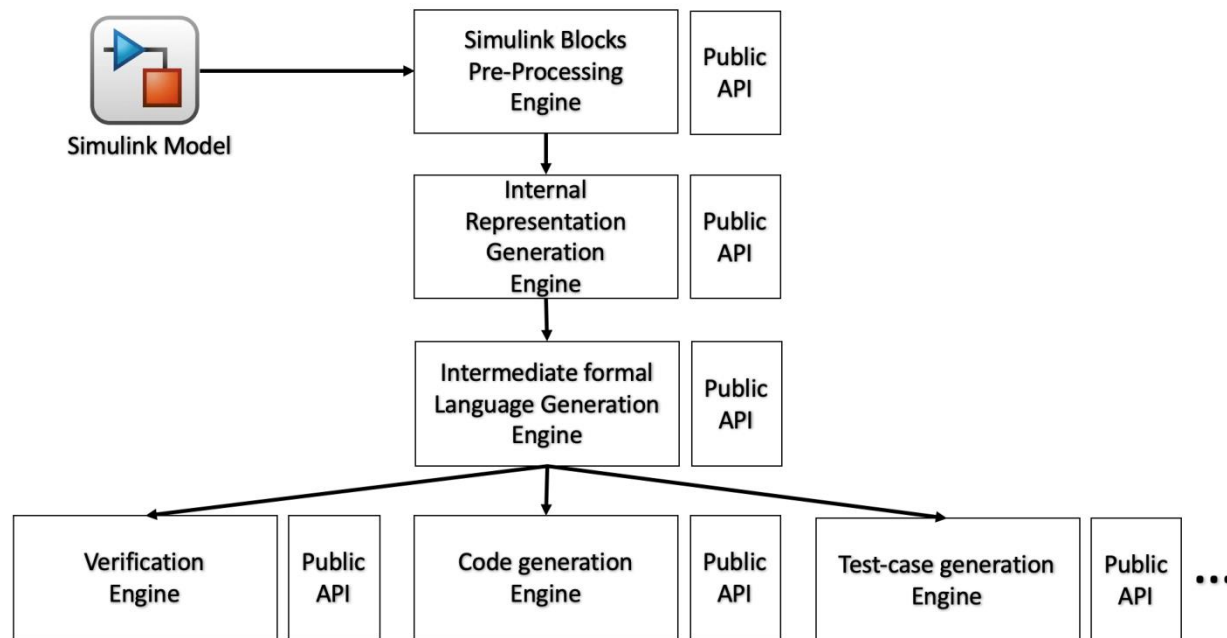
Integration on the target

- Steps:
 - Code generation:
 - Scade → C: KCG
 - ad hoc → scheduling + C
 - Test

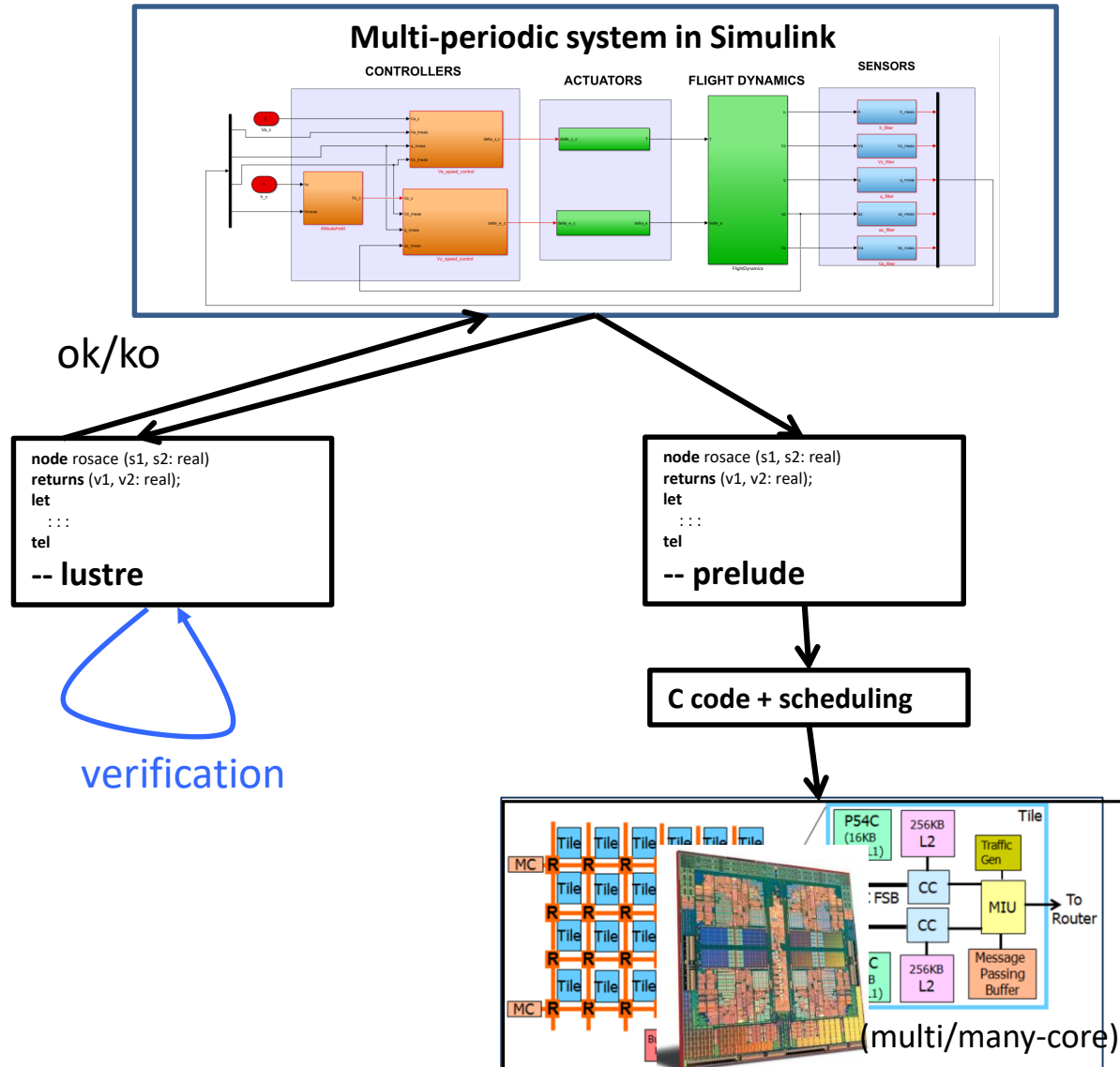
CoCoSim: what for?

❑ Open-source tool

- Simulink → Lustre/Prelude
- Verification capabilities – model checking with Kind2, Jkind, Zustre...
- Test case generation (MC-DC and mutation based testing)
- Customizable and configurable (any user can easily add their features)



CoCoSim for multi-periodic systems

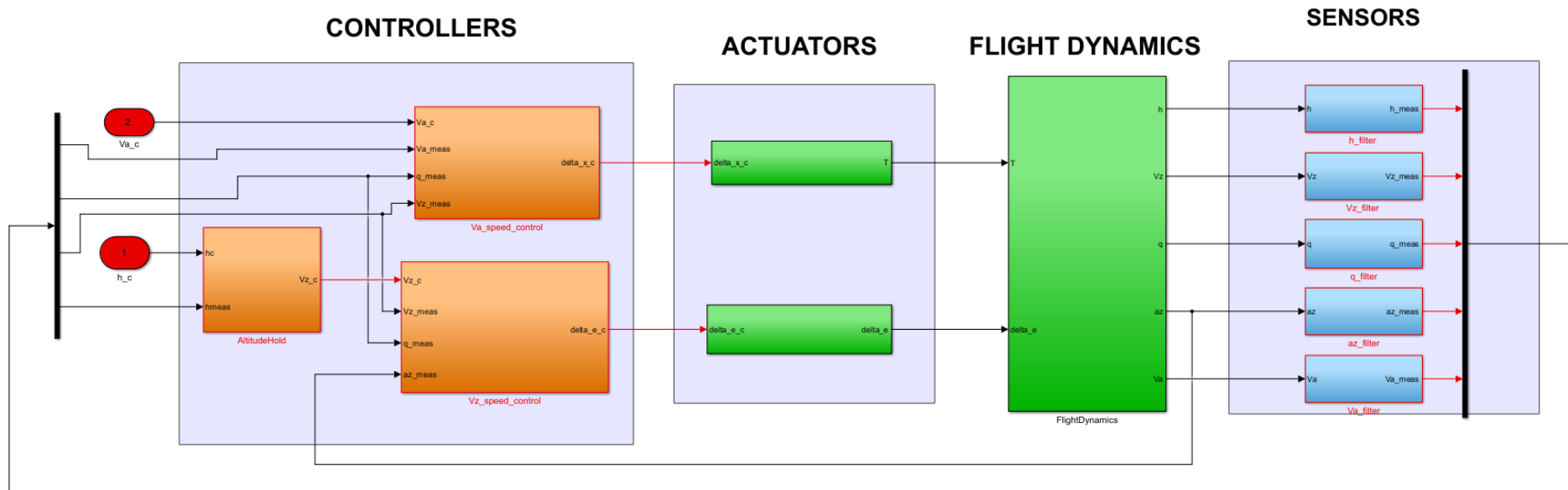


Outline

- ❑ Introduction
- ❑ **Reminder on Simulink and synchronous languages**
 - **Simulink**
 - **Lustre**
 - **Prelude**
- ❑ CoCoSim for multi-periodic systems
- ❑ Two open source use cases: ROSACE and Space shuttle AOCS
- ❑ Conclusion

Simulink – reminder

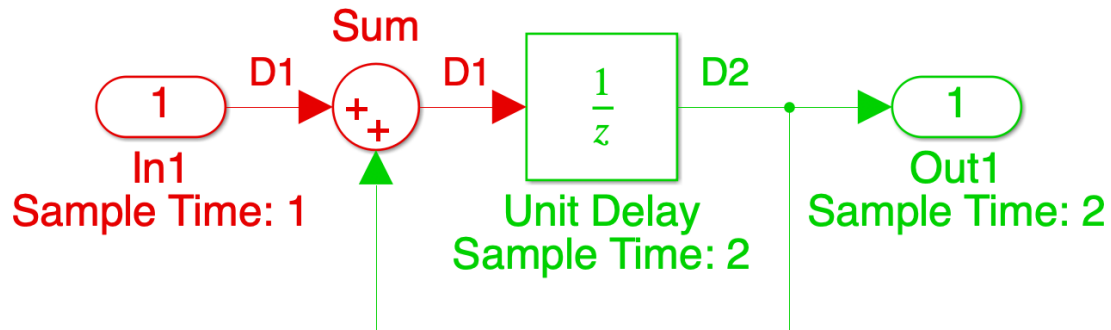
- ❑ Simulink is a graphical, dataflow programming environment for modeling and simulating dynamical systems.
- ❑ Simulink supports both discrete and continuous time semantic.
- ❑ A discrete Simulink model runs on a fixed time step defined with a period π and initial offset θ .



Multi-periodic systems in Simulink

- ❑ Any block b_i is set with a sample time $D = (\pi_i, \theta_i)$
- ❑ Updates only at times $k\pi_i + \theta_i$ for $k \in \mathbb{N}$, whereas, it remains constant during the intervals $[k\pi_i + \theta_i, (k+1)\pi_i + \theta_i]$

Example (Implicit handling)



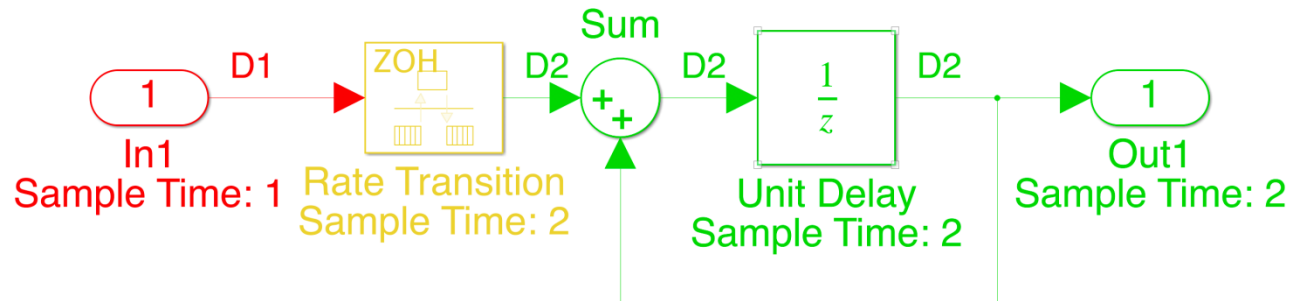
2 sample time domains: $D1=(1s, 0s)$ and $D2 = (2s, 0s)$

t	0	1	2	3	4	5
In1	1	1	1	1	1	1
Out1	0	0	1	1	2	2

Multi-periodic systems in Simulink

- ❑ By default, Simulink introduces implicit rate transition blocks
- ❑ User can force Simulink to reject models with unspecified data transfers between different rates

Example (Explicit handling)



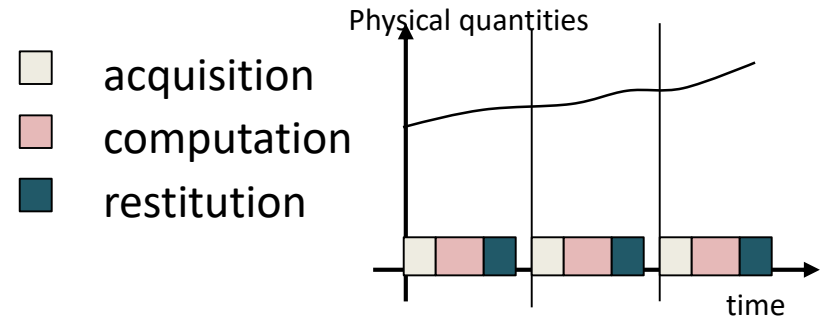
2 sample time domains: $D1=(1s, 0s)$ and $D2 = (2s, 0s)$

Same behaviour

t	0	1	2	3	4	5
In1	1	1	1	1	1	1
Out1	0	0	1	1	2	2

Reminder on synchronous languages

- Developed by engineers and formalised by researchers in the 80s
 - Esterel, Lustre (Scade), Signal, Lucid synchronise
- Synchronous hypothesis: computations are done during logical instant and must be finished before the next logical instant.
 - ⇒ the system behaves in « 0 time »
 - ⇒ simplification of the behaviour
 - time = succession of instants
 - composability of programs
- Sequential generated code
- Specification of multi-periodic systems not easy

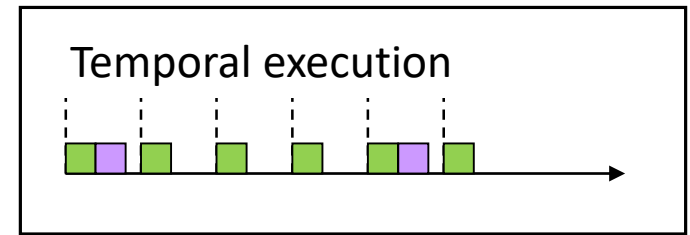
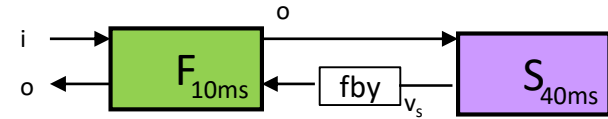


Example of assembly in Lustre

```

extern node F (i,j : int) returns (o:int);
extern node S (i : int) returns (o:int);
node multi_rate (i: int) returns (o: int)
var count, vs: int; clock4: bool;
let
  count=0 fby (count + 1);
  clock4=(count mod 4=0);
  vs=S(o when clock4);
  o=F(i, current (0 fby vs));
tel

```



Synchronous hypothesis

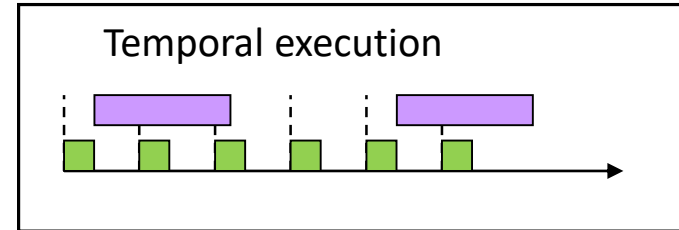
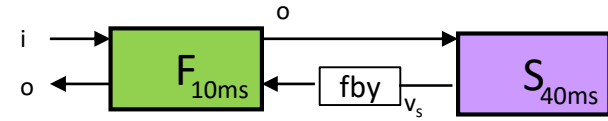
i	i ¹	i ²	i ³	i ⁴	i ⁵	i ⁶	...
count	0	1	2	3	4	5	...
count mod 4	0	1	2	3	0	1	...
clock4	true	false	false	false	true	false	...
o	o ¹ =F(i ¹ ,0)	o ² =F(i ² ,0)	o ³ =F(i ³ ,0)	o ⁴ =F(i ⁴ ,0)	o ⁵ =F(i ⁵ ,s ¹)	o ⁶ =F(i ⁶ ,s ¹)	...
o when clock4	o ¹				o ⁵		...
vs	s ¹ =S(o ¹)				s ² =S(o ⁵)		...
0 fby vs	0				s ¹		
current (0 fby vs)	0	0	0	0	s ¹	s ¹	

Same example in Prelude

```

imported node F (i,j : int) returns (o:int) wcet 5;
imported node S (i : int) returns (o:int) wcet 15;
node multi_rate (i: int rate (10,0)) returns (o: int)
var vs: int;
let
  vs=S(o/^4 );
  o=F(i, (0 fby vs) *^4);
tel

```

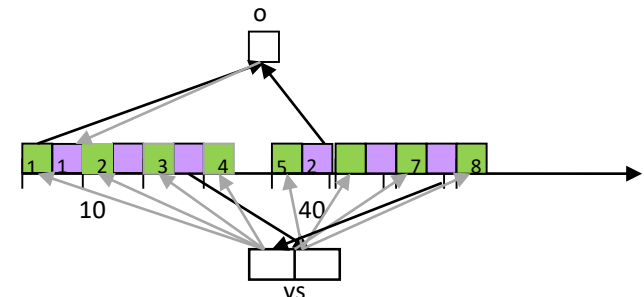


Relaxed synchronous hypothesis (Curic 2005)

i	i ¹	i ²	i ³	i ⁴	i ⁵	i ⁶	...
o	o ¹ =F(i ¹ ,0)	o ² =F(i ² ,0)	o ³ =F(i ³ ,0)	o ⁴ =F(i ⁴ ,0)	o ⁵ =F(i ⁵ ,s ¹)	o ⁶ =F(i ⁶ ,s ¹)	...
o/^4	o ¹				o ⁵		...
vs	s ¹ =S(o ¹)				s ² =S(o ⁵)		...
(0 fby vs)^4	0	0	0	0	s ¹	s ¹	...

Communication protocol

- Extension of Sofronis et al (2006)
- Independent from the scheduling policy



Outline

- ❑ Introduction
- ❑ Reminder on Simulink and synchronous languages
- ❑ **CoCoSim for multi-periodic systems**
 - **Verification**
 - **Code generation**
- ❑ Two open source use cases: ROSACE and Space shuttle AOCS
- ❑ Conclusion

Clock encoding in Lustre

For $D1=(1s, 0s)$ and $D2 = (2s, 0s)$

$D1 = \text{make_clock}(1,0)$ and $D2 = \text{make_clock}(2,0)$

where

node `make_clock` (`period` , `offset` : `int`)

returns (`clk` : `bool`)

var `count` : `int` ;

let

`count = ((period - offset) -> (pre (count) + 1) mod period ;`

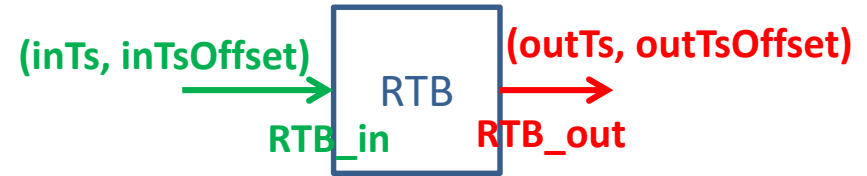
`clk = (count = 0);`

tel

t	0	1	2	3	4	5
<code>make_clock(1,0)</code>	true	true	true	true	true	true
<code>make_clock(2,0)</code>	true	false	true	false	true	false

Encoding of Simulink rate transitions in Lustre

```
C_in = make_clock (inTs, inTsOffset );  
C_out = make_clock (outTs, outTsOffset );
```



❑ From fast to slow: $\text{outTs} > \text{InTs}$ (ZOH block)

```
RTB_tmp = merge C_in RTB_in (( dft -> pre RTB_tmp ) when not C_in );  
RTB_out = RTB_tmp when C_out ;
```

❑ From slow to fast: $\text{outTs} < \text{InTs}$ (1/z block)

```
RTB_tmp = merge C_in ( dft -> pre RTB_in )(( dft -> pre RTB_tmp ) when not C_in );  
RTB_out = RTB_tmp when C_out ;
```

❑ Verification on standard Lustre

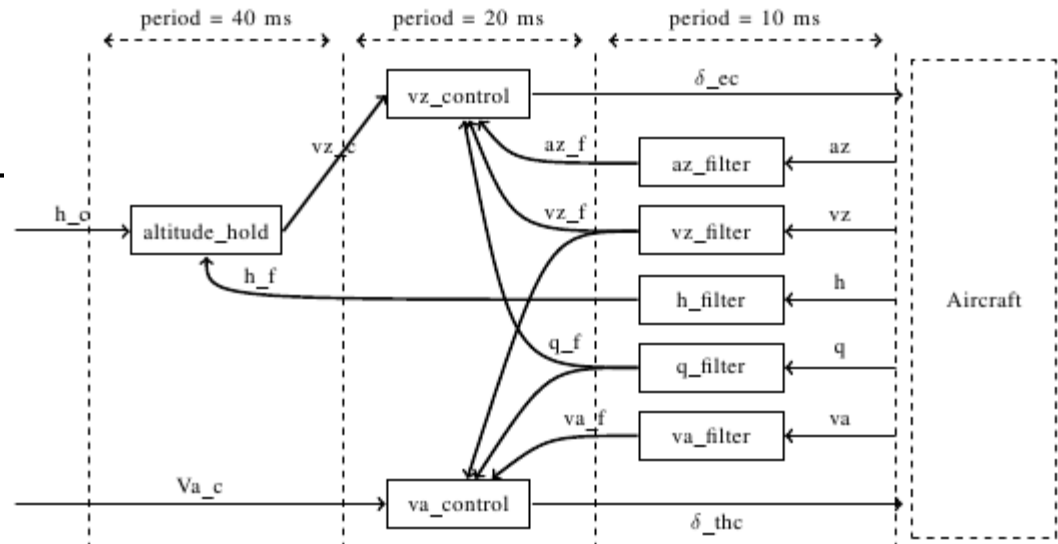
- Kind2: supports k-induction, IC3/PDR as well as on-the-fly invariant generation. Supported SMT solvers: CVC4, Z3, Yices.
- JKind: similar to Kind2 developed at Rockwell Collins.
- Zustre: based on Horn encoding describing the transition relation. SMT solvers: Z3.

Prelude – multi-periodic language

□ Synchronous language

```

imported node h_filter (h : real)
returns (h_f : real) wcet 25;
...
node rosace (h_c : real rate(100,0) ;
              Va_c : real rate(100,0) )
returns ( delta_x_c , delta_e_c )
var vz_c, va, az, q, vz , va_f, vz_f,
    az_f , q_f : real;
let
    va_f = va_filter(va/^2) ;
    delta_x_c = va_speed_control(Va_c/^20 , va_f/^2 , q_f/^2 , vz_f/^2) ;
    vz_f = vz_filter(vz/^2) ;
    delta_e_c = vz_speed_control( vz_c , vz_f/^2 , q_f/^2 , az_f/^2) ;
    az_f = az_filter(az/^2) ;
    h_f = h_filter(h/^2) ;
    q_f = q_filter(q/^2) ;
    vz_c = altitude_hold(h_c/^20 , h_f/^2) ;
    (va, az, q, vz , h) = aircraft_dynamics( (41814.0000000000 fby delta_x_c)^4 ,
                                              (0.0120000000 fby delta_e_c)^4 ) ;
tel
    
```



Outline

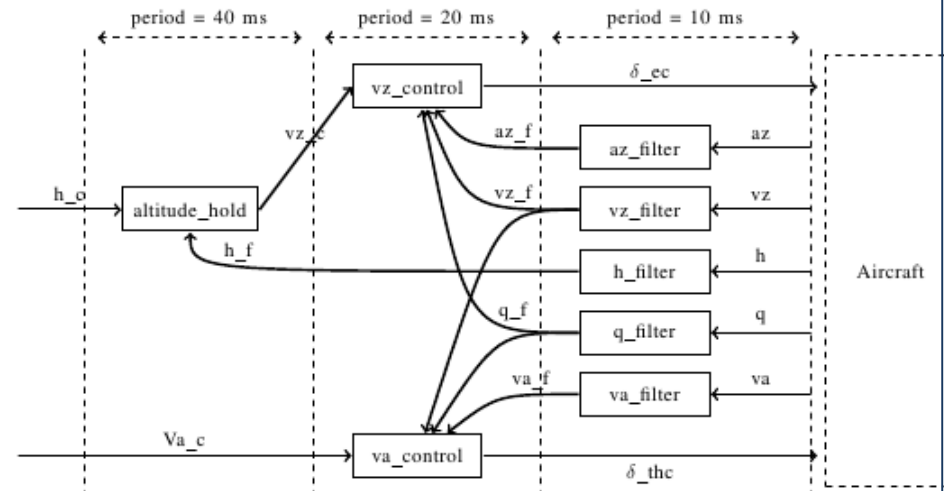
- ☐ Introduction
- ☐ Reminder on Simulink and synchronous languages
- ☐ CoCoSim for multi-periodic systems
- ☐ **Two open source use cases: ROSACE and Space shuttle AOCS**
- ☐ Conclusion

Two open source use cases – I

❑ ROSACE https://svn.onera.fr/schedmcore/branches/ROSACE_CaseStudy

❑ Available on the repository

- Simulink code
- C code
- Lustre/Prelude code
- Giotto
- Python script checker

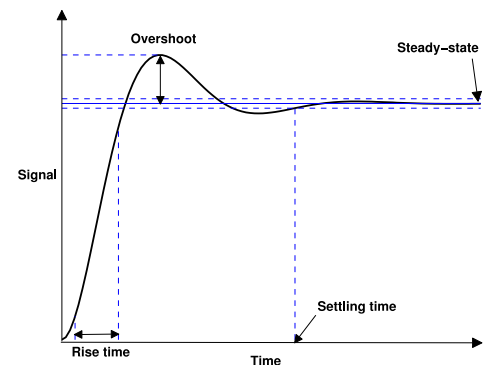


Longitudinal motion of a medium-range civil aircraft in *en-route* phase

- *Cruise*: maintains a constant altitude h and a constant airspeed Va
- *Change of cruise level* subphases

Performance requirements

- **Settling time** : time required to settle within 5% of the steady-state value
- **Overshoot** : maximum value attained minus the steady-state value
- **Rise time** : time to rise from 10% to 90% of the steady-state value

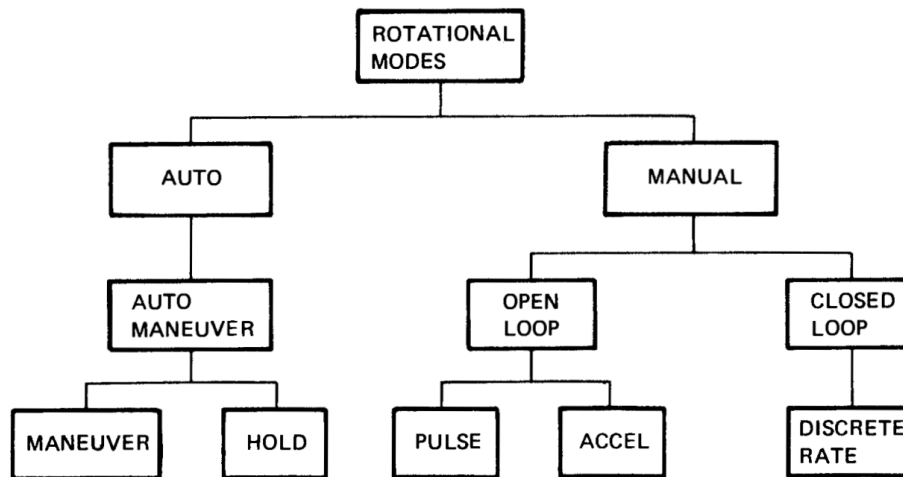


Two open source use cases – II

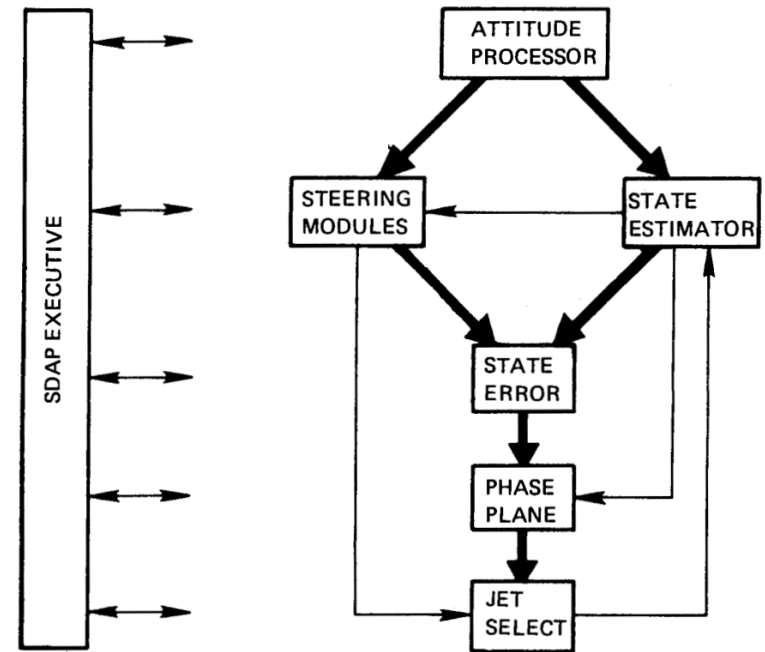
- ❑ Attitude and Orbital Control System (AOCS) of the Space Shuttle

<https://github.com/coco-team/spaceshuttle>

- ❑ Available in the repository: Simulink, Lustre and properties



Modding possibilities



Auto-maneuver
modules architecture

Example of safety properties

Req ID	Requirement
Req_p63_1	The two types of thrusters may not be used simultaneously
Req_p19_1	If the hand controller is deflected in any axis, the SDAP automatically switches to manual mode
Req_p19_5	When the maneuver mode is changed from manual to auto, if the bypass flag is ON, it is set to OFF and the auto-maneuver initialization flag is set to ON.
Req_p27_1	Auto Maneuver tests the rotation angle rotation angle delta theta against two numerical criteria. If rotation_angle_delta_theta is larger than $y = \text{SCALARBIAS} + 2 * \text{Deadband}$, the module places itself in the maneuver mode; if rotation_angle_delta_theta is less than $x = \text{SCALARBIAS} + \text{Deadband}$, the hold mode results.

Conclusion

- ❑ Open source development tool for control/command systems
- ❑ Provide verification and code generation

- ❑ Future works
 - Use in projects, e.g. H2020 PULSAR project
 - Extension to offer Monte-Carlo Tree search Rémi Delmas, Thomas Loquen, Josep Boada-Bauxell, Mathieu Carton: An Evaluation of Monte-Carlo Tree Search for Property Falsification on Hybrid Flight Control Laws. NSV@CAV 2019: 45-59
 - Interface with hybrid (continuous + discrete) verification tools
 - Dedicated code generation for neural network

Thanks for your attention