A SystemC/Simulink Co-Simulation Framework for Continuous/Discrete-Events Simulation

F. Bouchhima[1], M. Brière[1], G. Nicolescu[1], M. Abid[2], E. M. Aboulhamid[3]

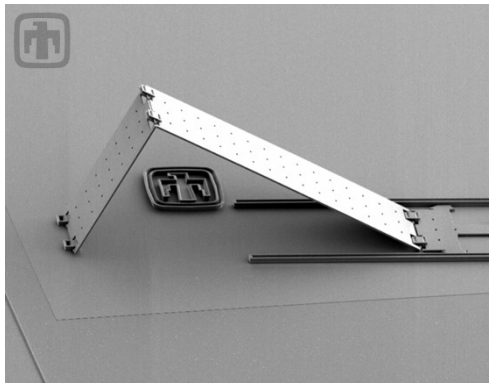
[1]Ecole Polytechnique de Montréal, Montréal, Canada

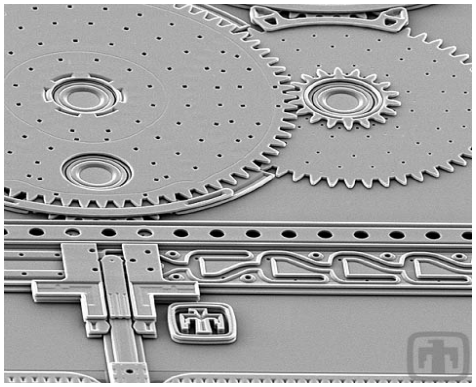
[2]National Engineer School of Sfax, Sfax, Tunisia

[3]Université de Montréal, Montréal, Canada

Continuous/Discrete systems

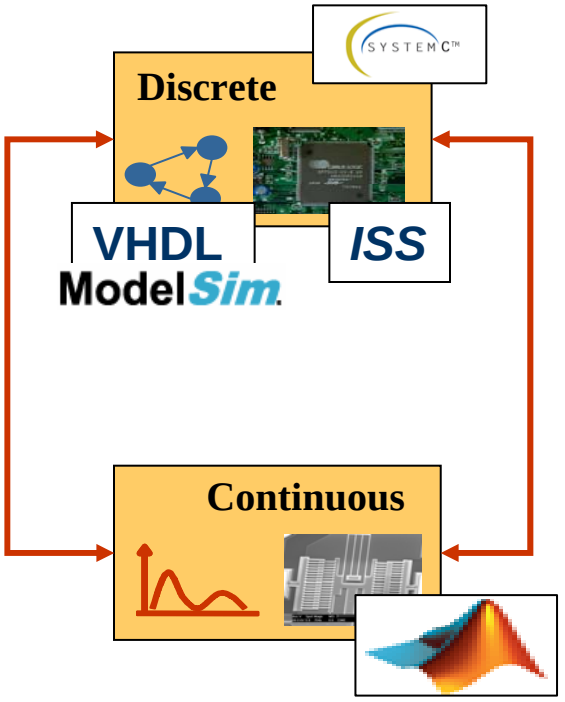
- Applications to various domains
  - Defense, medical, communications, automotive, …
- Main characteristics
  - Complexity, Heterogeneity
- Main design challenges
  - Global specification and validation





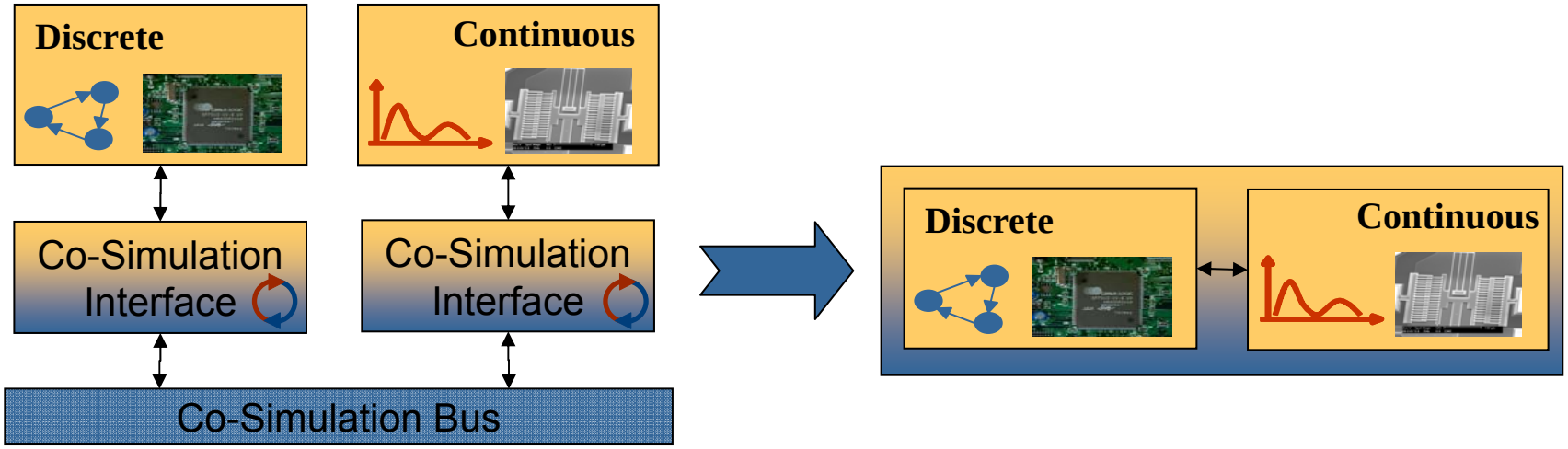
Continuous/Discrete systems design

- Collaboration between different teams
- Incremental refinements through different abstraction levels with specific execution models
- Validation requires joint execution of heterogeneous execution models
  - Co-Simulation technique



Challenges for Continuous/Discrete co-simulation

- Defining new tools facilitating cooperation between different teams
  - Exploiting powerful existing tools (Simulink, SystemC, …)
  - Taking into account implementation choices
  - Enabling easy specification, automatic generation for co-simulation interfaces



Contributions and objectives

- Fasten design of execution models for continuous/discrete co-simulation
- Definition of continuous/discrete framework: CODIS (**CO**ntinuous/**DI**screte **S**imulation)
- Find applications to illustrate the co-simulation framework powerful

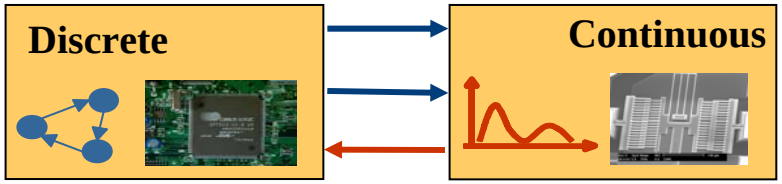
Outline

- Global simulation for continuous/discrete systems
- Synchronization models for continuous/discrete systems co-simulation
- CODIS framework
- Conclusion

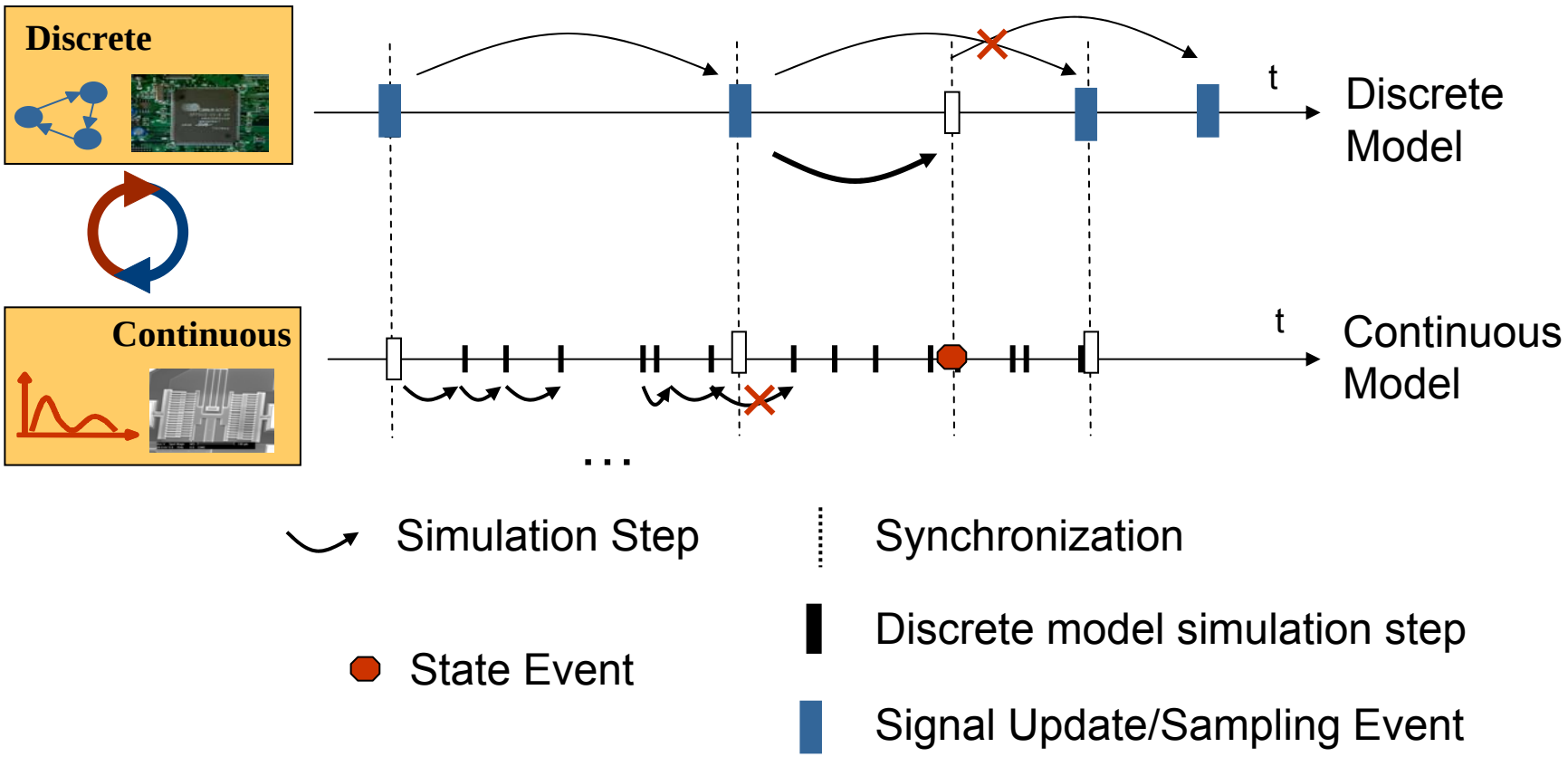
Continuous/Discrete systems
- heterogeneous execution models -

Concept / Model	Time	Communication means	Processes activation rules
Discrete	Advances discretely	Set of events	Processes are sensitive to events
Continuous	Advances by integration steps (IS)	Piecewise-Continuous signals	Processes are executed at each IS

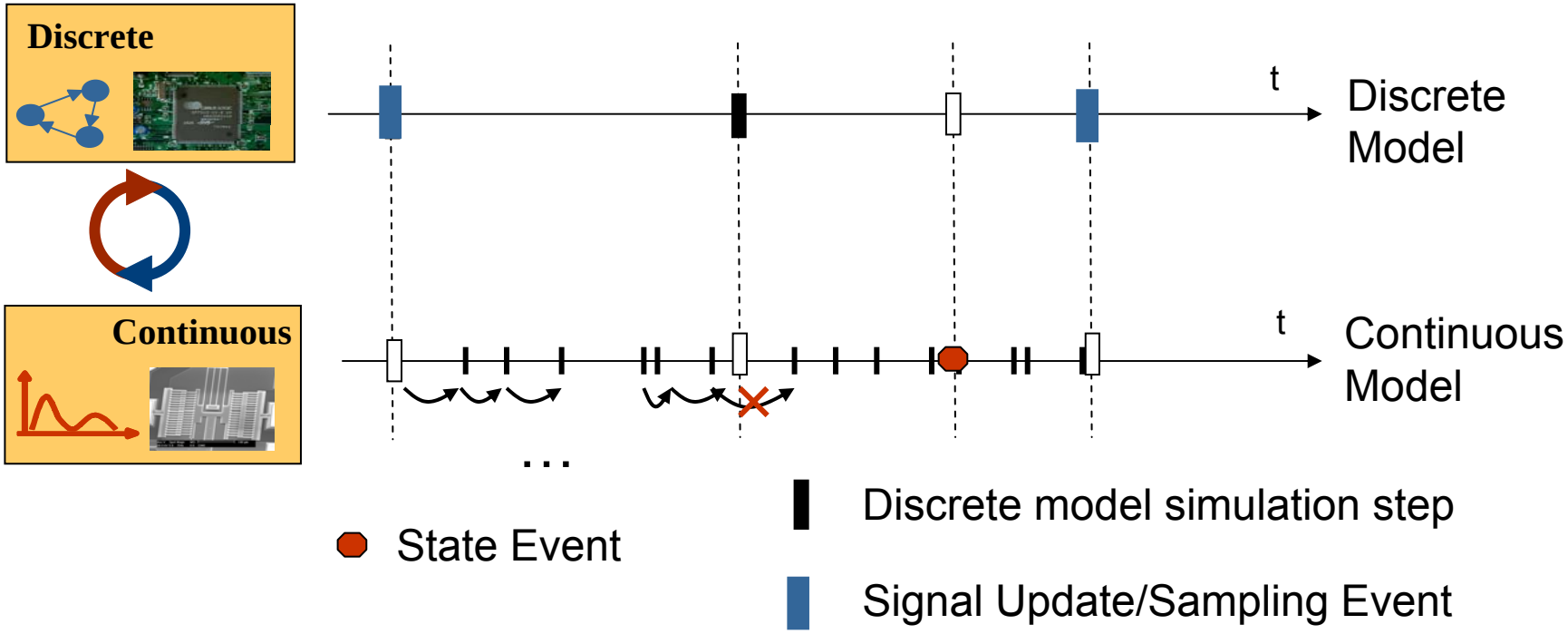
- Events exchanged between the two models
  - Sampling events
  - Update signal events
  - State events



Continuous/Discrete systems
- heterogeneous execution models -



Continuous/Discrete systems
- synchronization models -



Full synchronization mode

Each discrete step and/or state event occurrence

Predictable events mode

Each update/sampling events and/or state event occurrence

Unpredictable events mode (≡ events-driven mode)

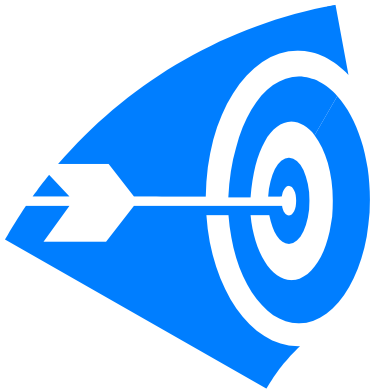
Each update/sampling events and/or state event occurrence

Continuous/Discrete systems - synchronization models -

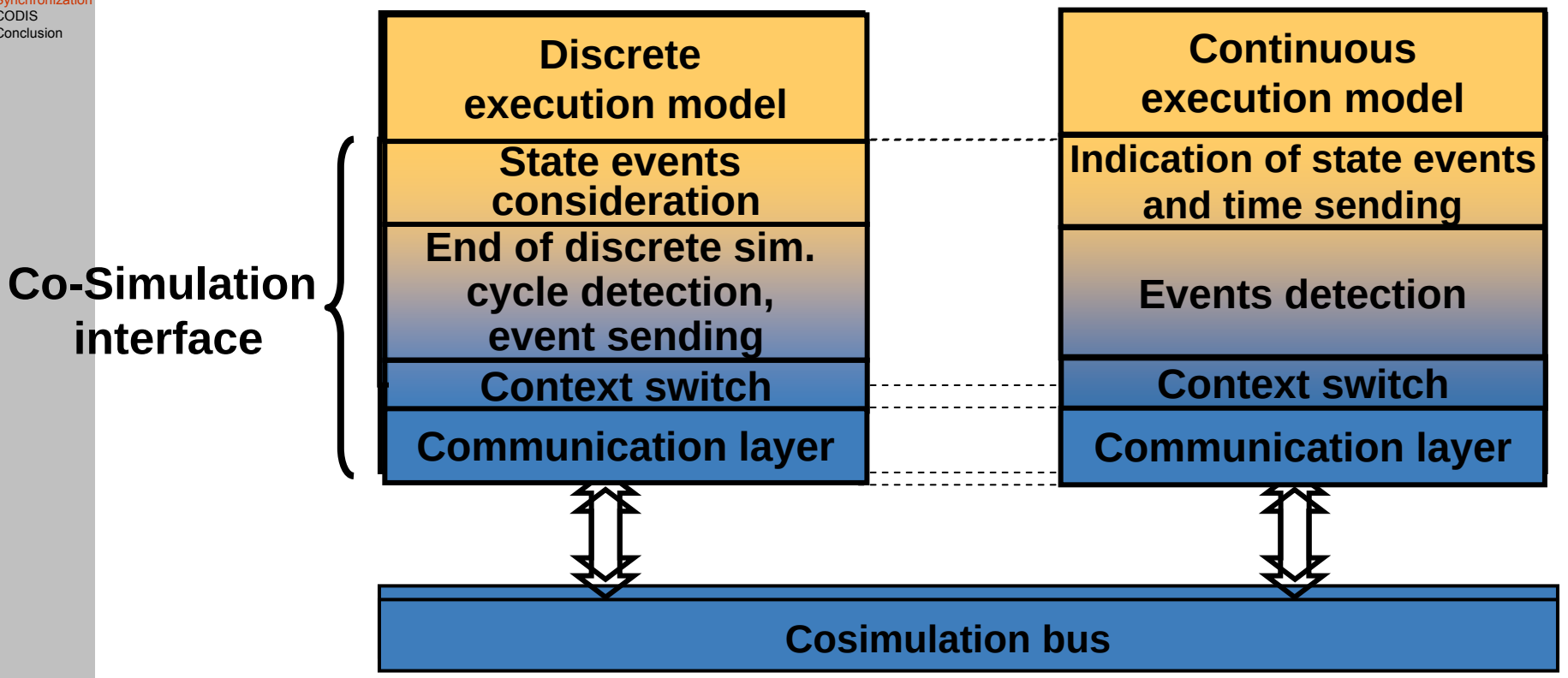
Synchronization model	Synchronization step	Advantages	Inconvenient
Full synchronization mode	Each discrete step and/or state event occurrence	General	Synchronization overhead
Predictable events mode	Each update/sampling events and/or state event occurrence	Improve performances	The prediction of update/sample events is required
Unpredictable events mode (≡ events-driven mode)	Each update/sampling events and/or state event occurrence	Non-periodic update/sample events	Rollback may be required

Challenges for accurate global execution

- Considering the state events generated by the continuous simulator
- Detection of the next event of the discrete simulator (scheduled event)
- Detection of the end of the discrete simulation cycle and the time step sending



Generic architecture for continuous/discrete co-simulation



Outline

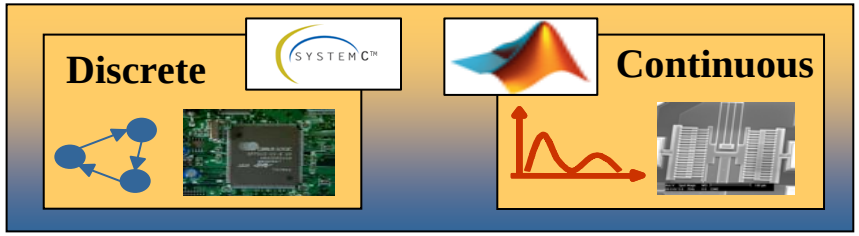
- Global simulation for continuous/discrete systems
- Synchronization models for continuous/discrete systems co-simulation
- CODIS framework
- Conclusion

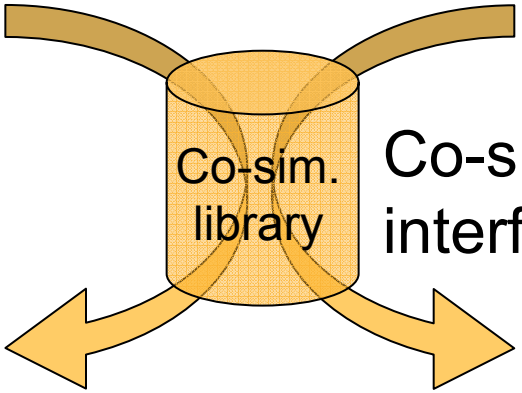
CODIS framework

- **CO**ntinuous/**DI**screte **S**imulation
- Implementation of the synchronization models
- First prototype enabling Simulink/SystemC simulation
- Integrate easily discrete SystemC components
- Integrate easily continuous Simulink components

CODIS framework - simulation flow -

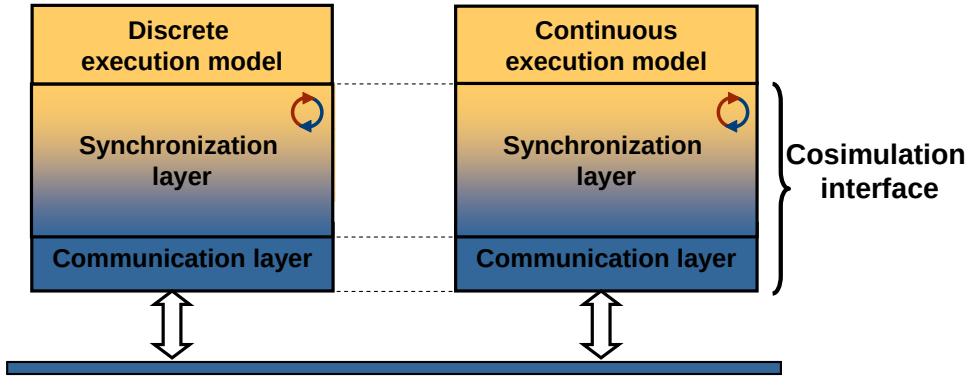
Input flow



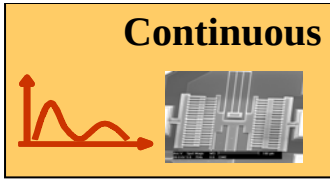


Co-simulation
interfaces generation

Output flow

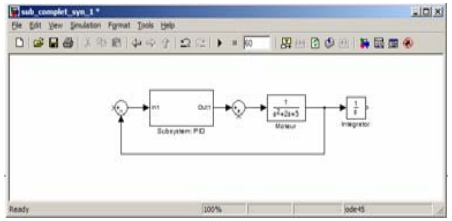


Generation of continuous model interfaces

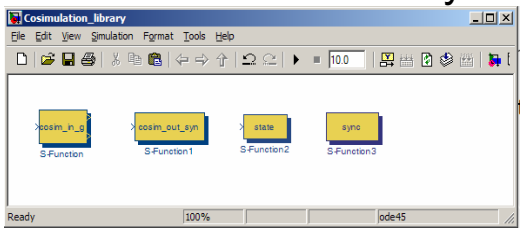


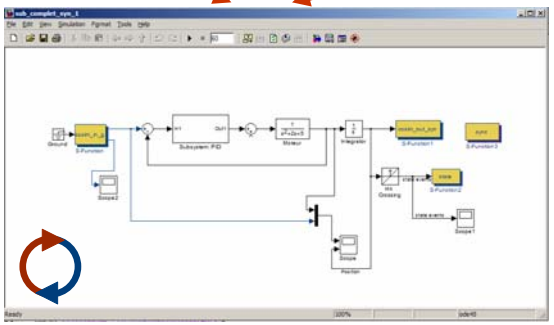
- Simulink co-simulation library blocks
  - State-event detection and signalling
  - Synchronization with sampling and update events
  - Integration step adjustment
  - Communication

Simulink input specification



Co-simulation library





Simulink specification with co-simulation interfaces

Generation of discrete model interfaces

- Based on a SystemC co-simulation library
  - State events management blocks
  - Communication blocks
- Automatic generation of co-simulation interfaces by a code generator that has as input user-defined parameters
  - Data types
  - Number and type of ports
  - Synchronization model

New tasks added in the SystemC scheduler

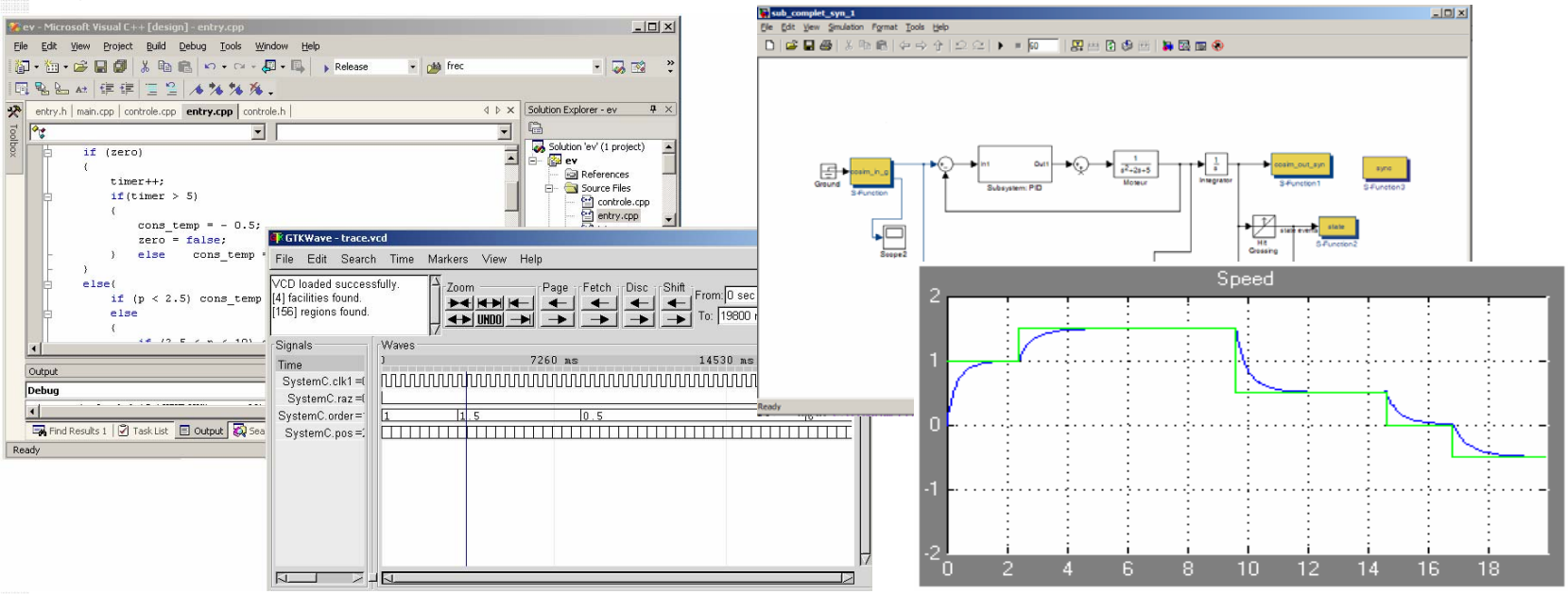
1. *Initialization Phase* – Execute all processes (except SC_CTHREADs) in an unspecified order.
2. *Evaluate Phase* – Select a process that is ready to run and resume its execution.
3. If there are still processes ready to run, go to step 2.
4. *Update Phase* – Execute any pending calls to update() resulting from request_update() calls made in step 2.
5. If there are pending delayed notifications, determine which processes are ready to run due to the delayed notifications and go to step 2.
6. **If Mode = FS then Send the next discrete time to Simulink and Switch context to Simulink,**
   Else
   If Mode = PE then send the next *signals update event* or *sampling event* time stamp and Switch context to Simulink.
   Else (mode = ED) if *signals update event* flag = "1" or *sampling event* then send the current time and Switch context to Simulink
7. **If state event then add a timed event with time stamp equal to the state event time stamp to the scheduler queue.**
8. If there are no more timed notifications, simulation is finished.

CODIS - COntinuous/DIscrete Simulation -

- SystemC/Simulink accurate simulation
  - Easy integration, generic library elements

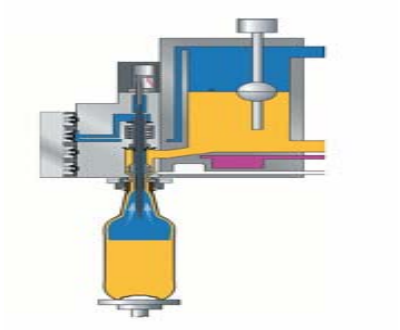
SystemC

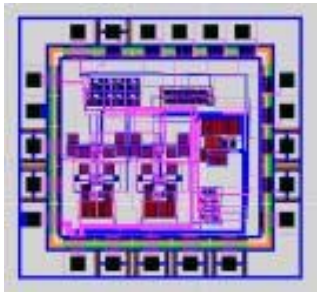
Simulink



CODIS - applications -

- Control applications
  - Bottle filling station, robot arm manipulator
- Mixed signal application
  - Σ/Δ converter
- Wireless application
  - Radar system





CODIS - application - bottle filling station

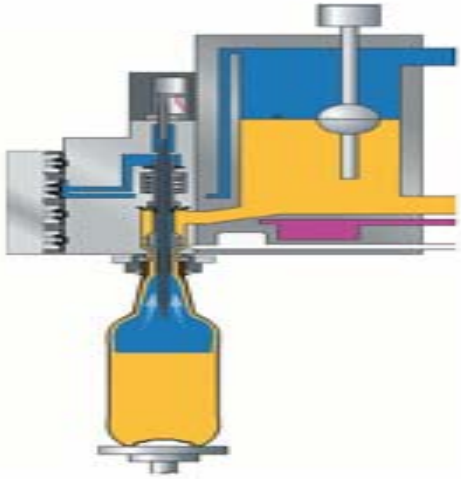
State event

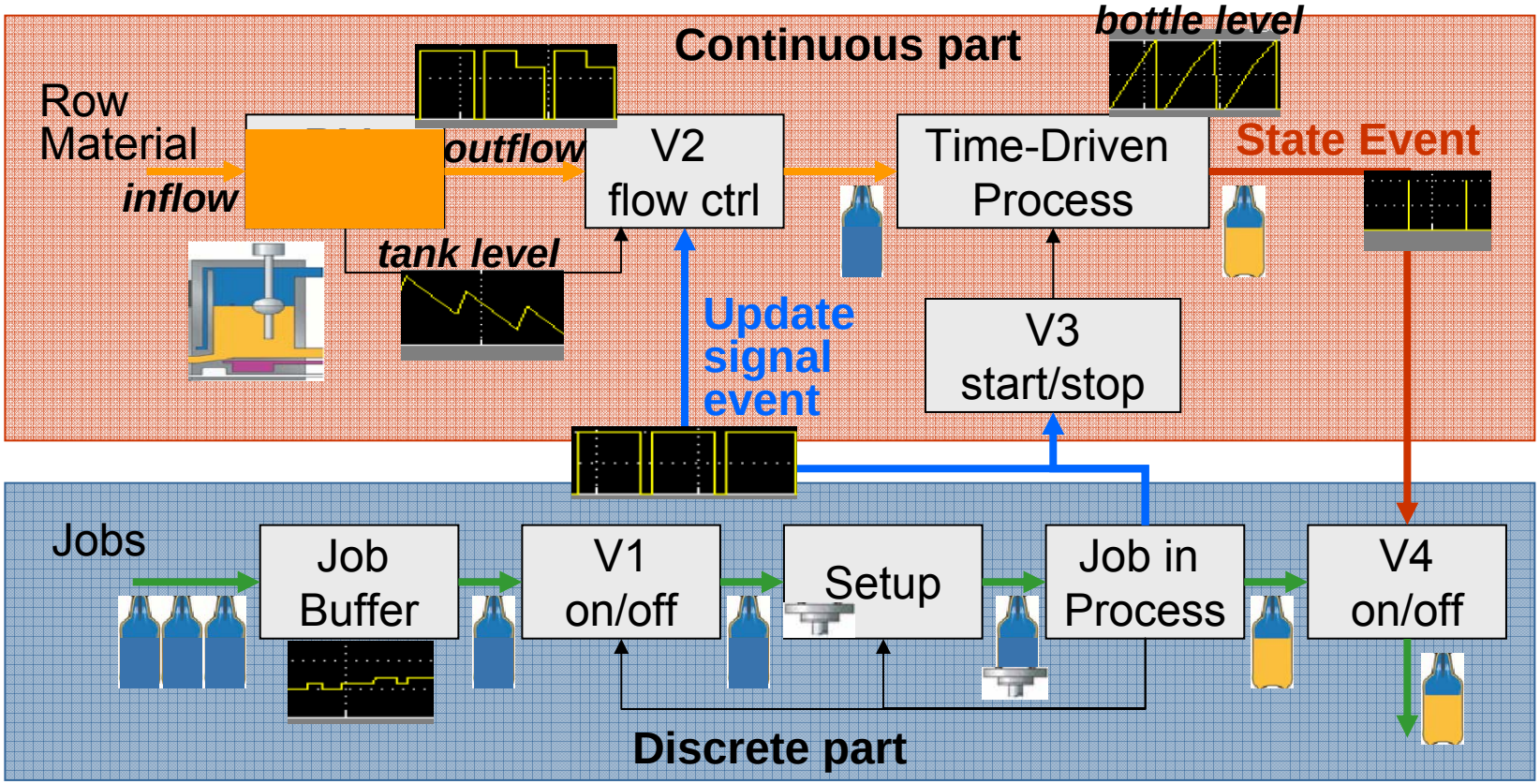
Update signal event

Row material flow

Job flow

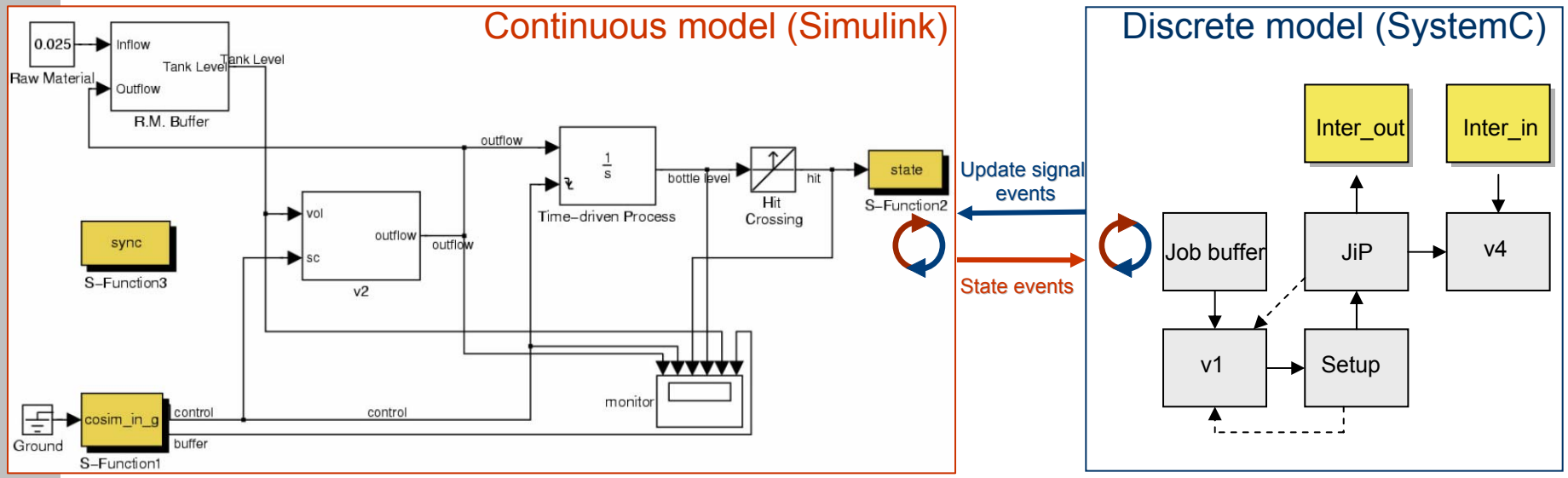
Internal control





CODIS - application - bottle filling station

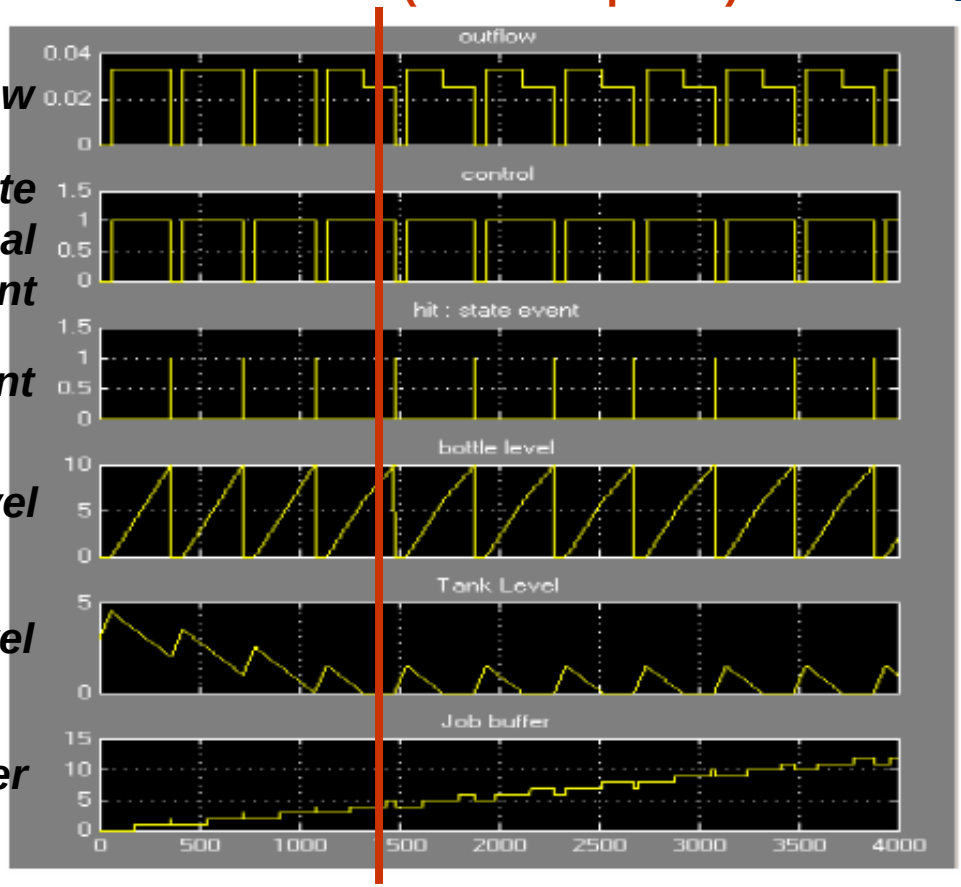
- Diagram presentation
  - Discrete part models the job flow using SystemC components
  - Continuous part models the raw material flow using Simulink components



CODIS - application - bottle filling station

- Simulation results
  - Full synchronization mode (state events and signals update events are not periodic)

Continuous model (Simulink probe)



outflow

Update signal event

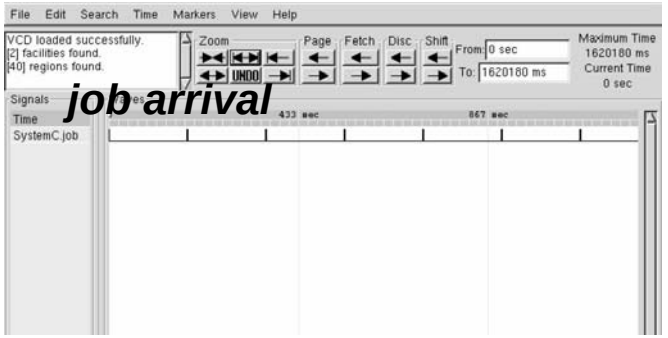
state event

bottle level

tank level

job buffer

Discrete model (SystemC *vcd* trace)



job arrival

Configuration
job arrival rate: 180 s
setup: 60 s
inflow rate: 0.025 l/s
max. outflow rate: 0.033 l/s
min. outflow rate: 0.025 l/s

Performances analysis

- Inter-simulators communication overhead
  - 20% of the total simulation time
- Overhead caused by the Simulink integration step adjustment
  - maximum 5% of total simulation time
- SystemC synchronization overhead
  - maximum 0.2% of the total simulation time

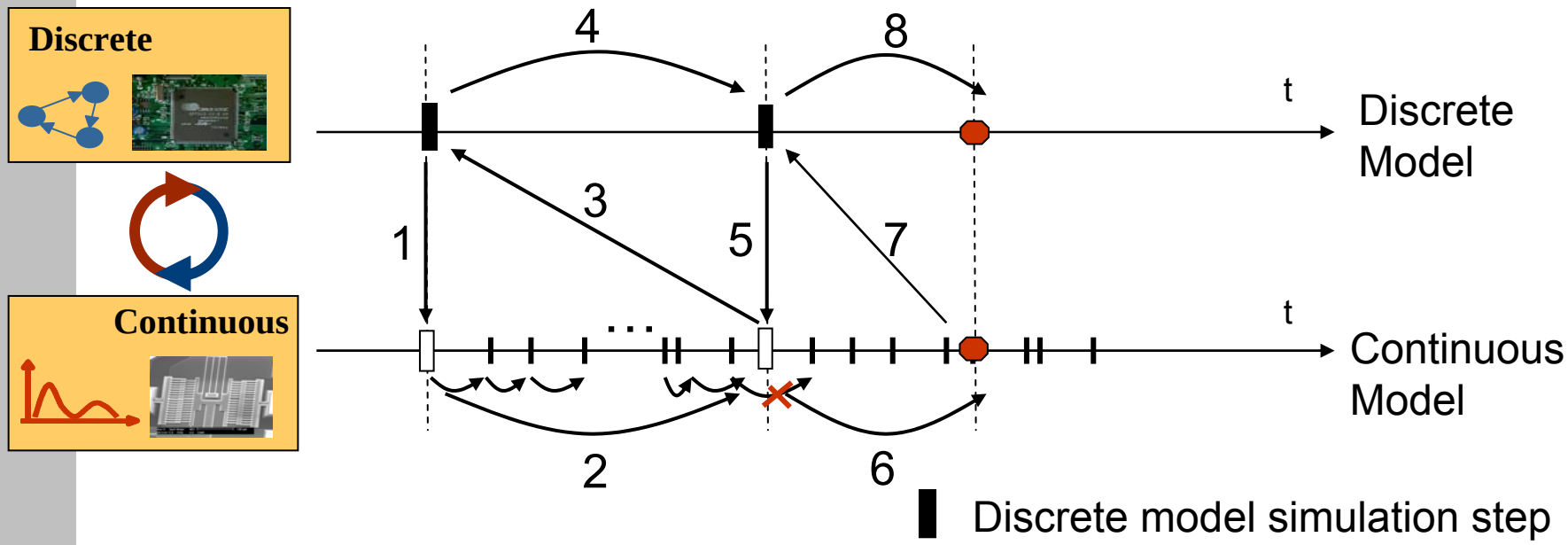
Conclusions

- The design of continuous/discrete systems requires co-simulation based validation
- Challenges for continuous/discrete co-simulation
  - Definition of global execution models respecting different synchronization models
  - Co-simulation interfaces automatic generation
- CODIS
  - Co-simulation framework enabling generation of global execution models
  - SystemC/Simulink co-simulation

BACKUP

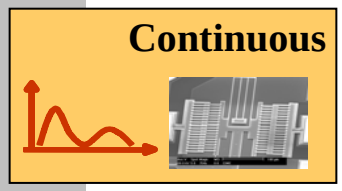
Continuous/Discrete systems
- synchronization models -

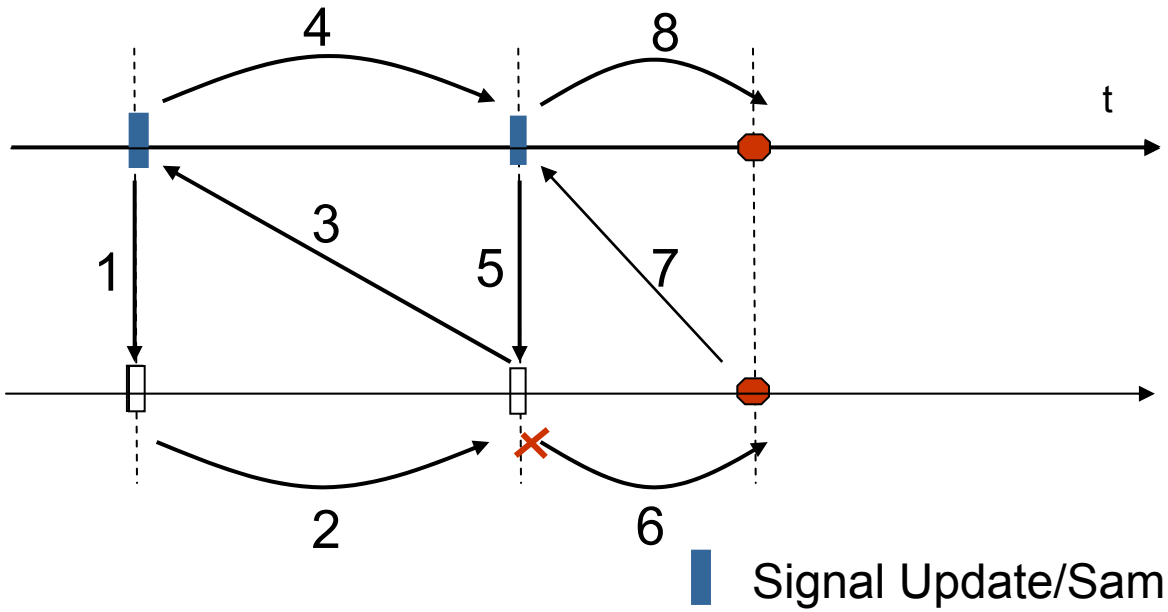
- Full synchronization mode
  - Synchronization – each discrete model step and/or state event occurrence
  - General
  - Synchronization overhead



Continuous/Discrete systems
- synchronization models -

- Prediction events mode
  - Synchronization – each update/sampling events and/or state event occurrence
  - ☺ Improve performances
  - ☹ Requires prediction for update/sampling events





Continuous/Discrete systems
- synchronization models -

- Unpredictable events mode
  - Synchronization – each update/sampling events and/or state event occurrence
  - Accept non-periodic update/sampling events, efficient when no state events occurs
  - Rollback for discrete model is required when state events occur