The Power Window Case Study in ModHel'X

Frédéric Boulanger Christophe Jacquet
Cécile Hardebolle Ayman Dogui

Supélec E3S – Department of Computer Science

Miami, September 29 2013

2013-09-20 15:18

Agenda

Introduction

Introduction

Software

Sensors

Networks

Control

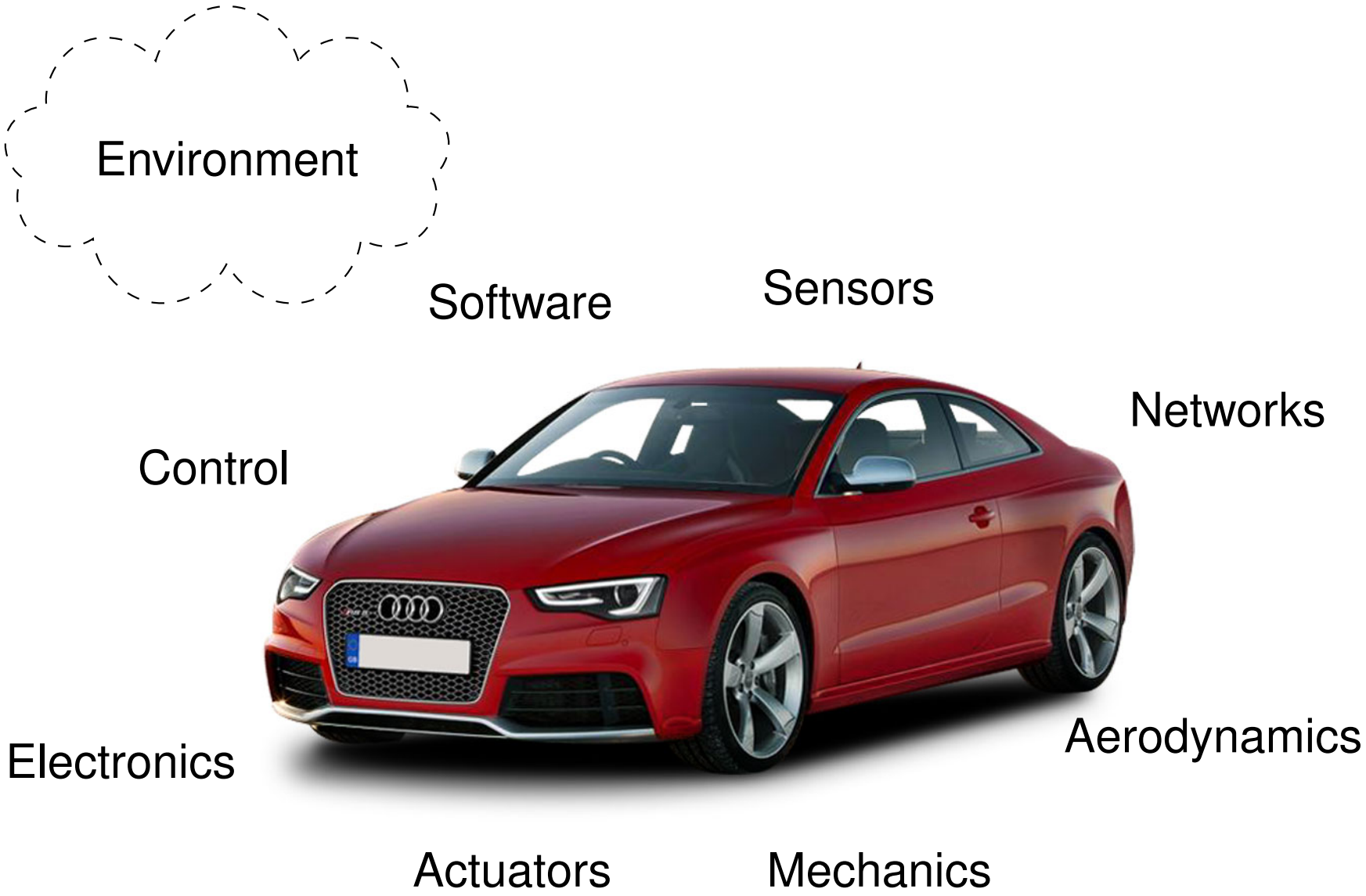
Electronics

Aerodynamics

Actuators

Mechanics

Introduction



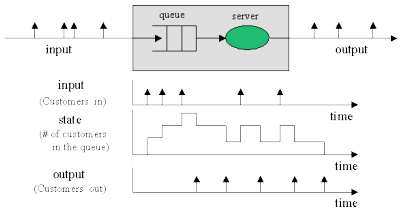
Introduction

Environment

$$\frac{dS_T}{dt} = r_T - S_T\left(\frac{\beta_{I_B} S_T I_B + \beta_{I_V} S_T I_V}{N_B + N_V}\right) - d_M S_T$$
$$\frac{dI_T}{dt} = S_T\left(\frac{\beta_{I_B} S_T I_B + \beta_{I_V} S_T I_V}{N_B + N_V}\right) - d_I I_T$$
$$\frac{dS_V}{dt} = r_V - S_V\left(\frac{\beta_{I_T} S_V I_T + \beta_{I_B} S_V I_B + \beta_{I_V} S_V I_V}{N_T + N_V + N_B}\right) - d_M S_V$$
$$\frac{dI_V}{dt} = S_V\left(\frac{\beta_{I_T} S_V I_T + \beta_{I_B} S_V I_B + \beta_{I_V} S_V I_V}{N_T + N_V + N_B}\right) - d_I I_V$$
$$\frac{dS_B}{dt} = r_B - S_B\left(\frac{\beta_{I_T} S_B I_T + \beta_{I_V} S_B I_V}{N_T + N_V}\right) - d_M S_B$$
$$\frac{dI_B}{dt} = S_B\left(\frac{\beta_{I_T} S_B I_T + \beta_{I_V} S_B I_V}{N_T + N_V}\right) - d_I I_B,$$

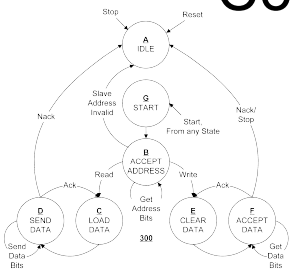
Software

Sensors



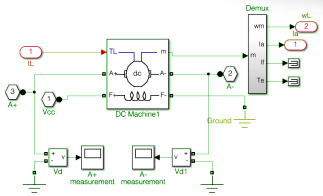
Networks

Control



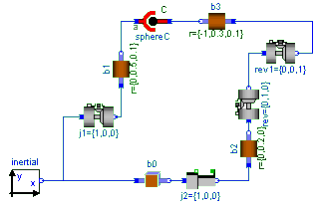
Electronics

Aerodynamics



Actuators

Mechanics



Semantics of heterogeneous models

- Semantics of modeling paradigms

$$\frac{dS_T}{dt} = r_T - S_T\left(\frac{\beta_{I_B} S_T I_B + \beta_{I_V} S_T I_V}{N_B + N_V}\right) - d_M S_T$$

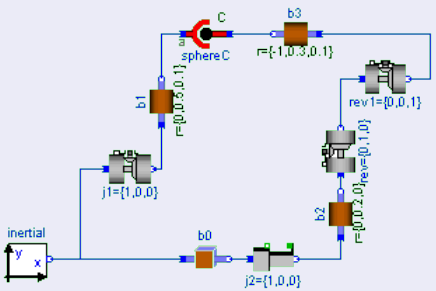
$$\frac{dI_T}{dt} = S_T\left(\frac{\beta_{I_B} S_T I_B + \beta_{I_V} S_T I_V}{N_B + N_V}\right) - d_I I_T$$

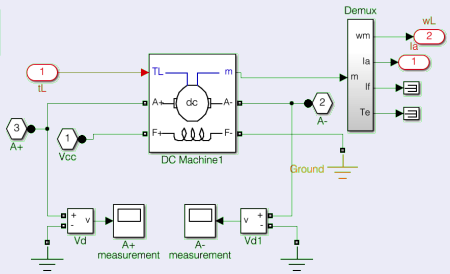
$$\frac{dS_V}{dt} = r_V - S_V\left(\frac{\beta_{I_T} S_V I_T + \beta_{I_B} S_V I_B + \beta_{I_V} S_V I_V}{N_T + N_V + N_B}\right) - d_M S_V$$

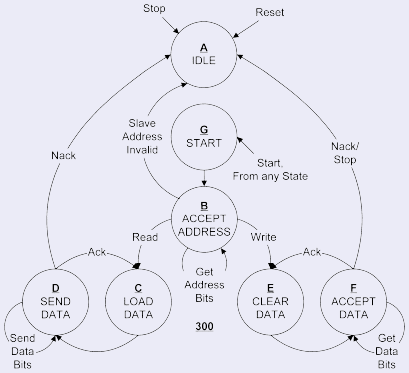
$$\frac{dI_V}{dt} = S_V\left(\frac{\beta_{I_T} S_V I_T + \beta_{I_B} S_V I_B + \beta_{I_V} S_V I_V}{N_T + N_V + N_B}\right) - d_I I_V$$

$$\frac{dS_B}{dt} = r_B - S_B\left(\frac{\beta_{I_T} S_B I_T + \beta_{I_V} S_B I_V}{N_T + N_V}\right) - d_M S_B$$

$$\frac{dI_B}{dt} = S_B\left(\frac{\beta_{I_T} S_B I_T + \beta_{I_V} S_B I_V}{N_T + N_V}\right) - d_I I_B,$$



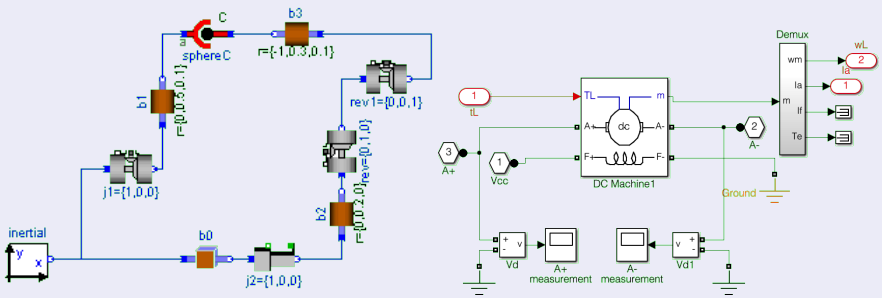


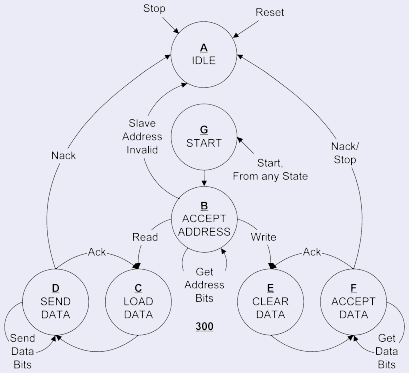


Semantics of heterogeneous models

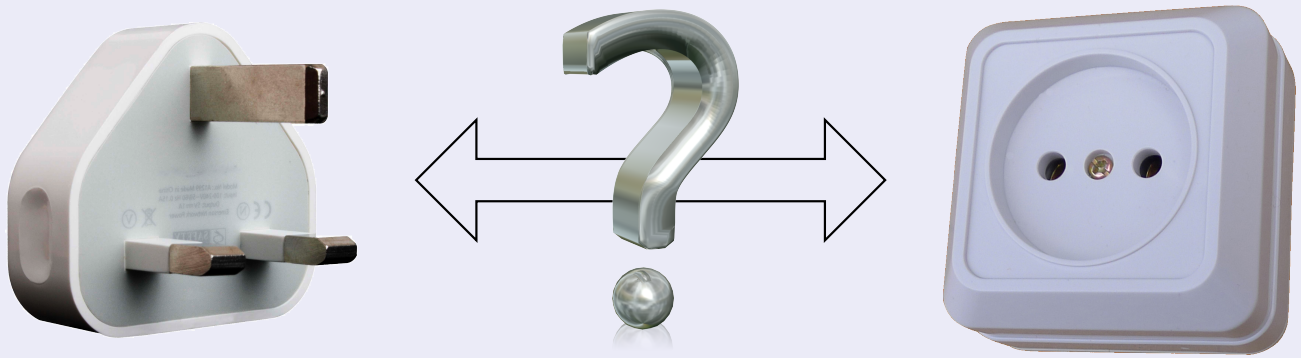
- Semantics of modeling paradigms

$$\frac{dS_T}{dt} = r_T - S_T\left(\frac{\beta_{I_B} S_T I_B + \beta_{I_V} S_T I_V}{N_B + N_V}\right) - d_M S_T$$
$$\frac{dI_T}{dt} = S_T\left(\frac{\beta_{I_B} S_T I_B + \beta_{I_V} S_T I_V}{N_B + N_V}\right) - d_I I_T$$
$$\frac{dS_V}{dt} = r_V - S_V\left(\frac{\beta_{I_T} S_V I_T + \beta_{I_B} S_V I_B + \beta_{I_V} S_V I_V}{N_T + N_V + N_B}\right) - d_M S_V$$
$$\frac{dI_V}{dt} = S_V\left(\frac{\beta_{I_T} S_V I_T + \beta_{I_B} S_V I_B + \beta_{I_V} S_V I_V}{N_T + N_V + N_B}\right) - d_I I_V$$
$$\frac{dS_B}{dt} = r_B - S_B\left(\frac{\beta_{I_T} S_B I_T + \beta_{I_V} S_B I_V}{N_T + N_V}\right) - d_M S_B$$
$$\frac{dI_B}{dt} = S_B\left(\frac{\beta_{I_T} S_B I_T + \beta_{I_V} S_B I_V}{N_T + N_V}\right) - d_I I_B,$$





- Semantic adaptation



ModHel'X: Generic Metamodel

Behavioral unit: Block, "black box" with an interface

ModHel'X: Generic Metamodel

Behavioral unit: Block, "black box" with an interface

Interface unit: Pin, to send and receive information

ModHel'X: Generic Metamodel

Behavioral unit: Block, "black box" with an interface

Interface unit: Pin, to send and receive information

Structure: Relations between pins, semantics defined by the MoC

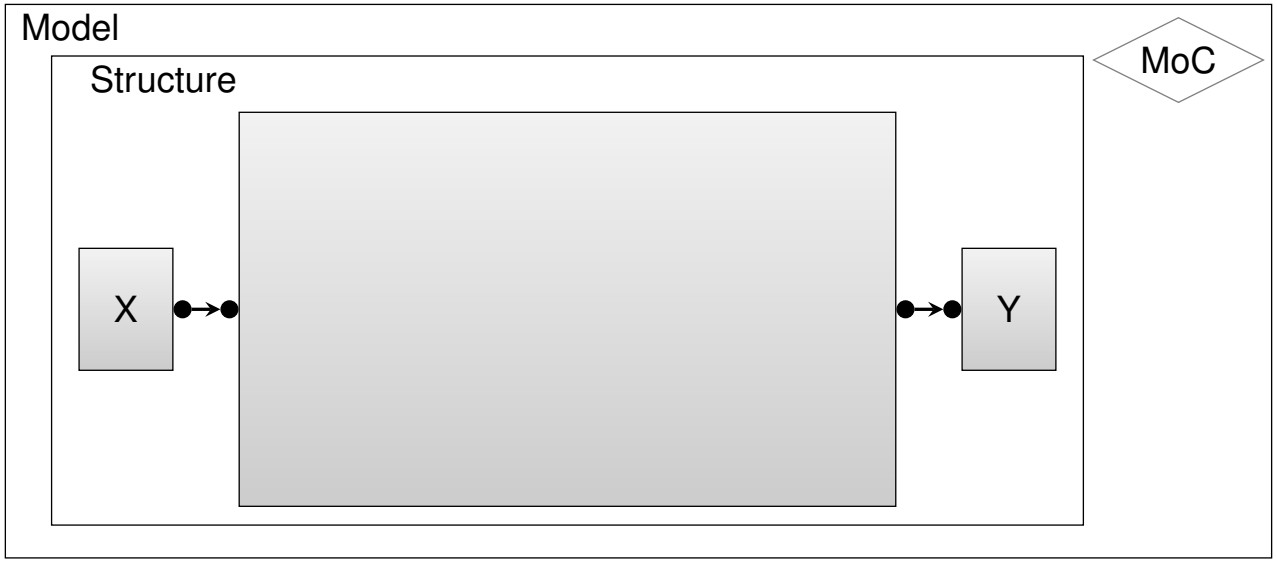
ModHel'X: Generic Metamodel

Behavioral unit: Block, "black box" with an interface

Interface unit: Pin, to send and receive information

Structure: Relations between pins, semantics defined by the MoC

Model: Model = structure + MoC



ModHel'X: Generic Metamodel

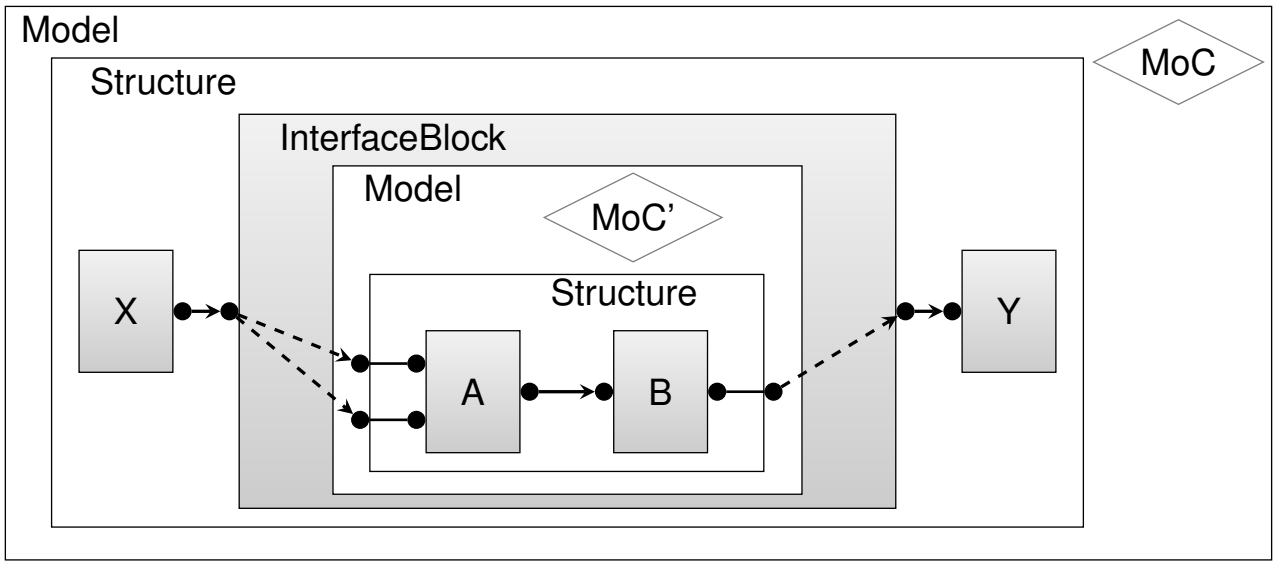
Behavioral unit: Block, "black box" with an interface

Interface unit: Pin, to send and receive information

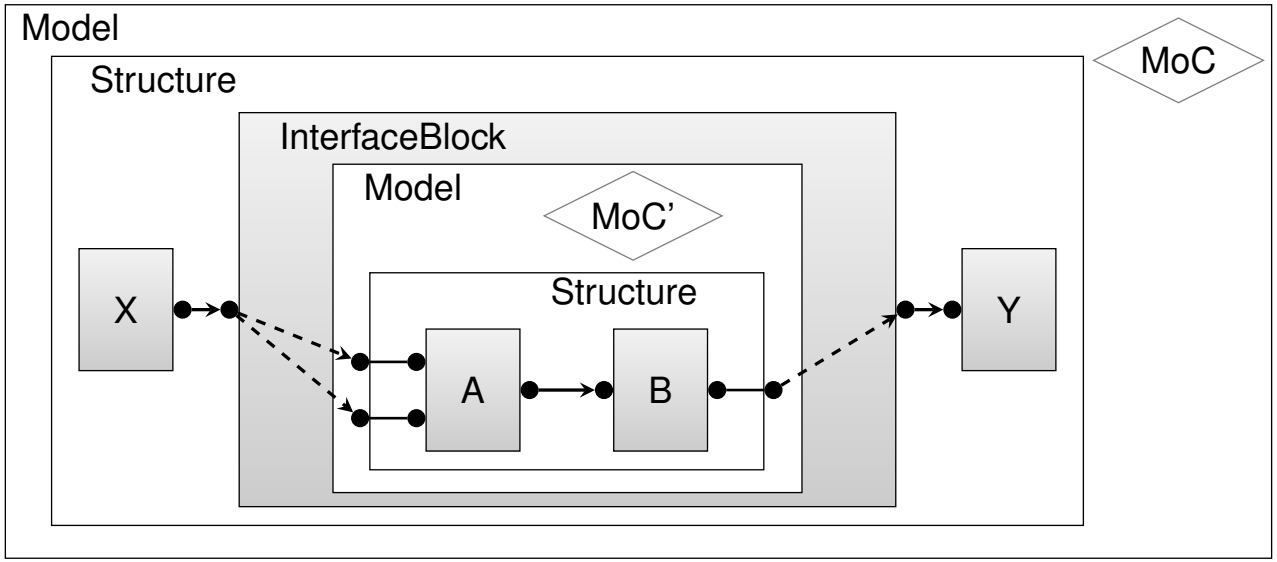
Structure: Relations between pins, semantics defined by the MoC

Model: Model = structure + MoC

InterfaceBlock: Hierarchical heterogeneity

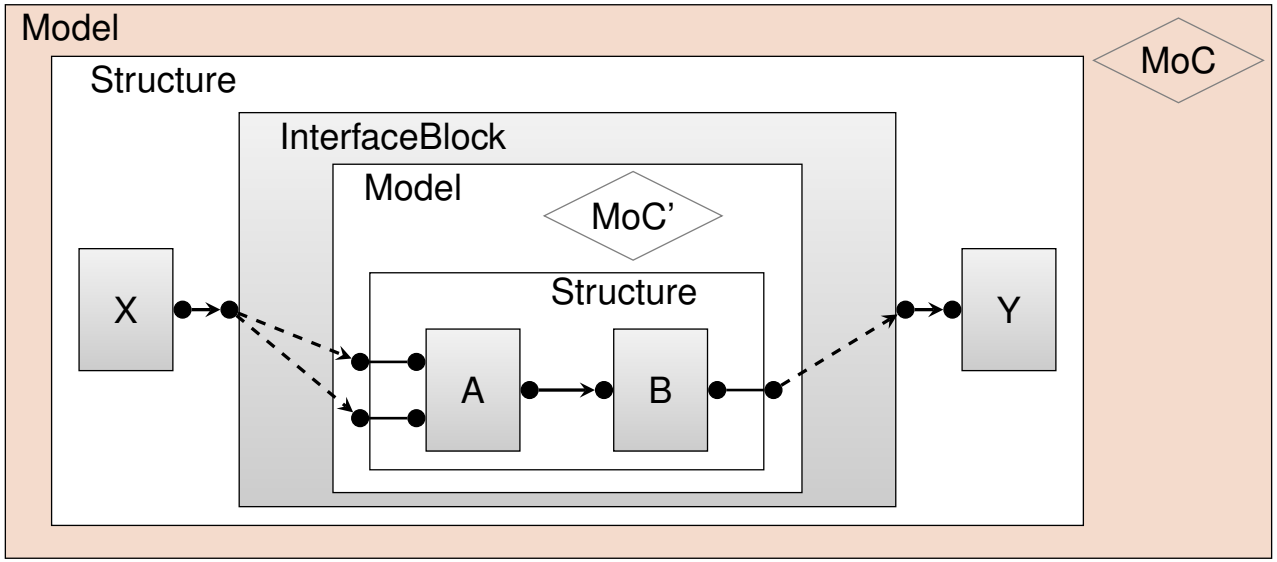


ModHel'X: Generic Execution Engine



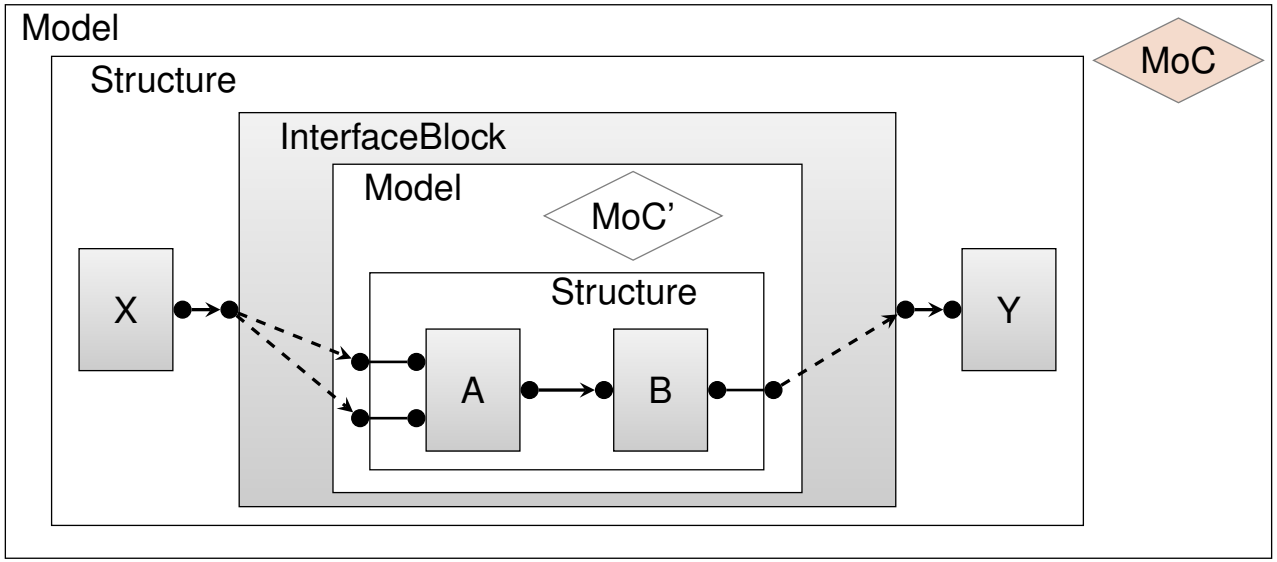
Execution of a model = series of observations (snaphots)

ModHel'X: Generic Execution Engine



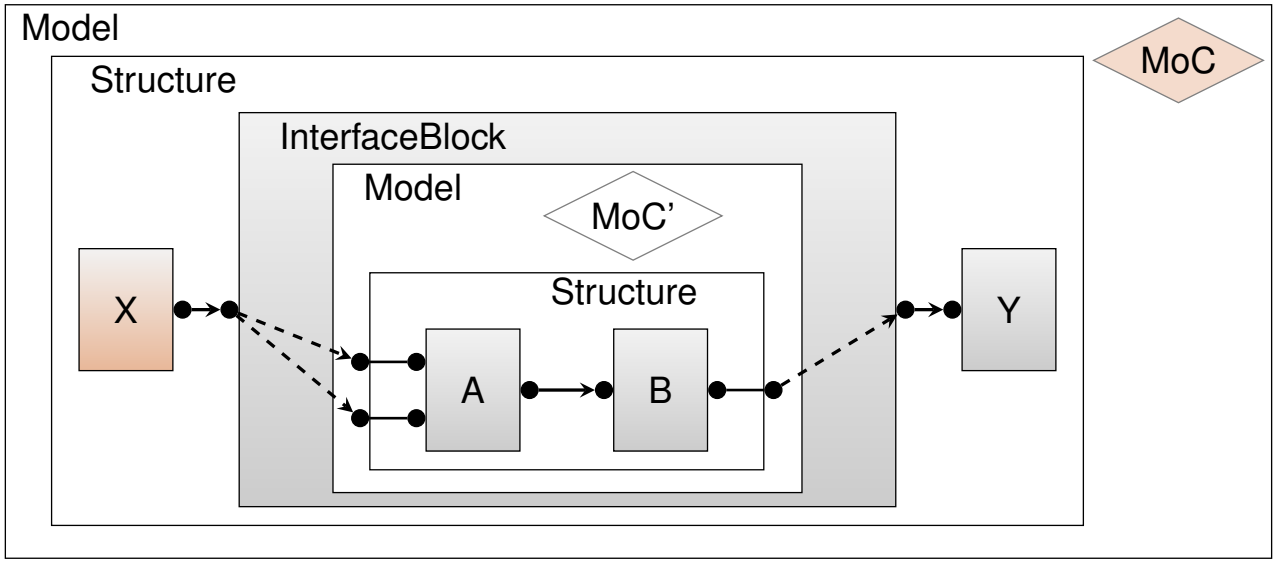
Snapshot = observation (update) of the root model

ModHel'X: Generic Execution Engine



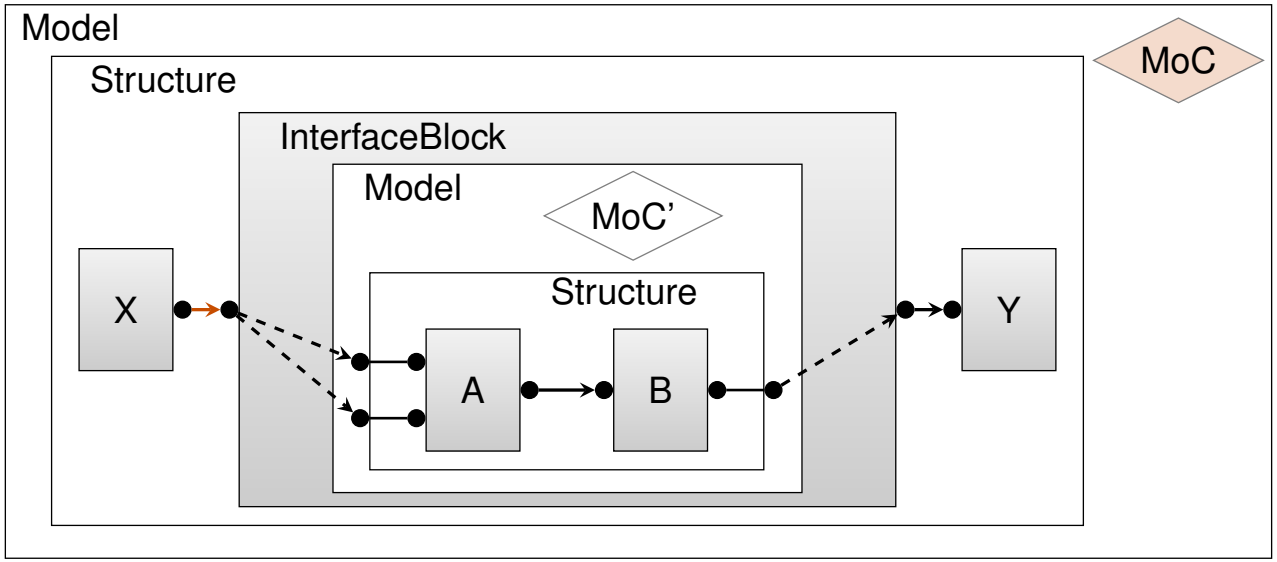
Control is given to the MoC of the root model

ModHel'X: Generic Execution Engine



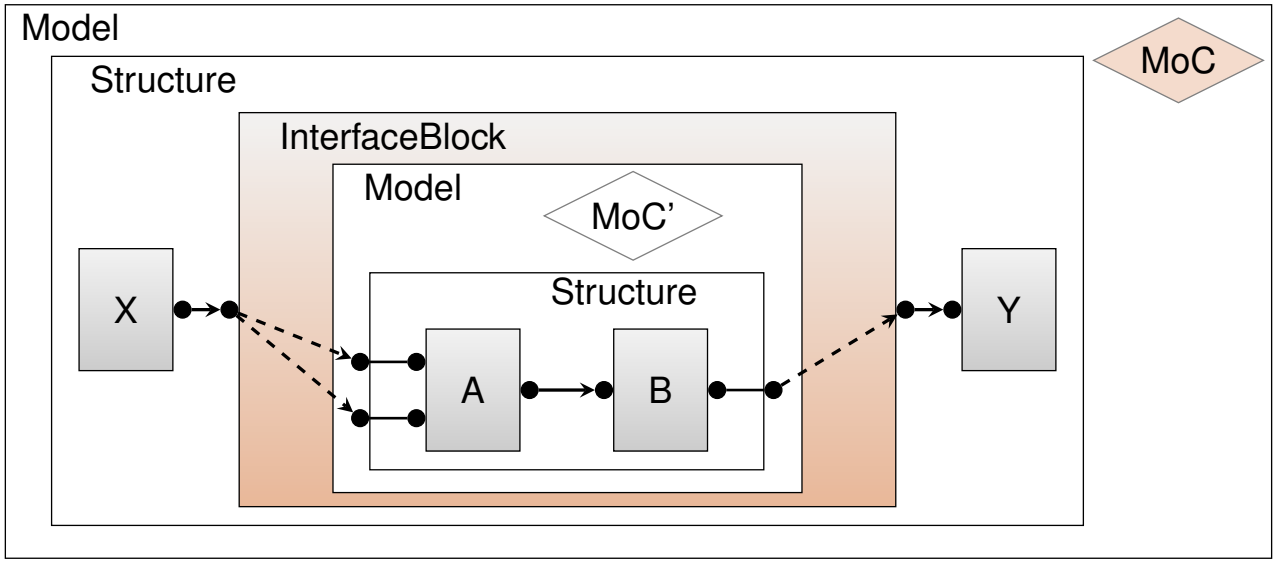
Which schedules block X for update

ModHel'X: Generic Execution Engine



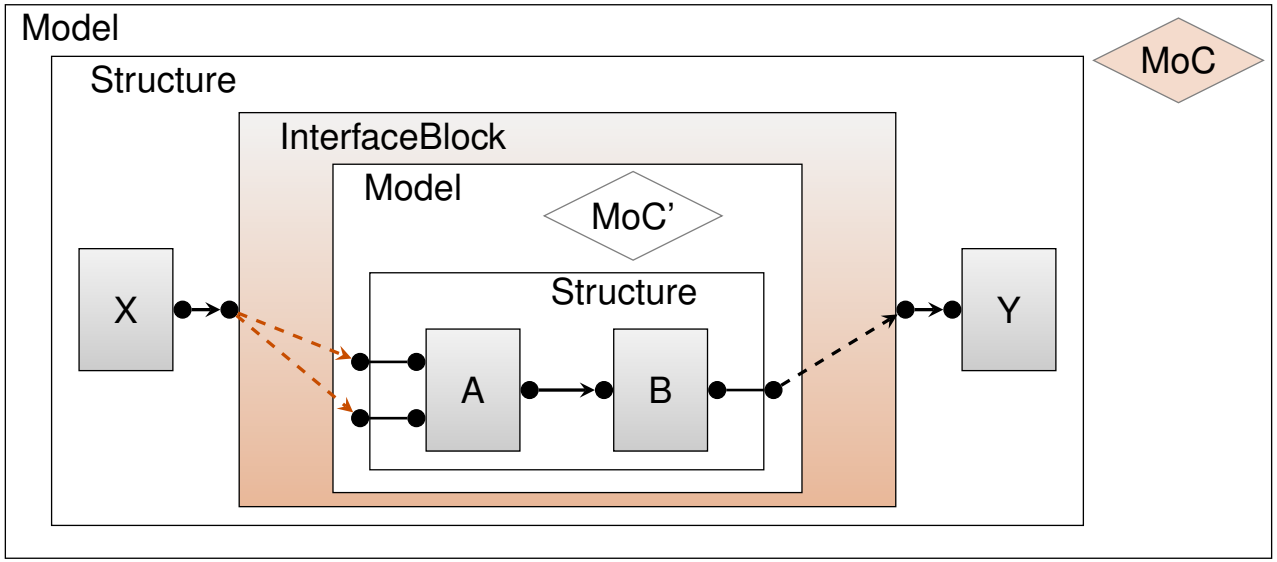
Propagates outputs of block X according to the relations

ModHel'X: Generic Execution Engine



Schedules the interface block for update

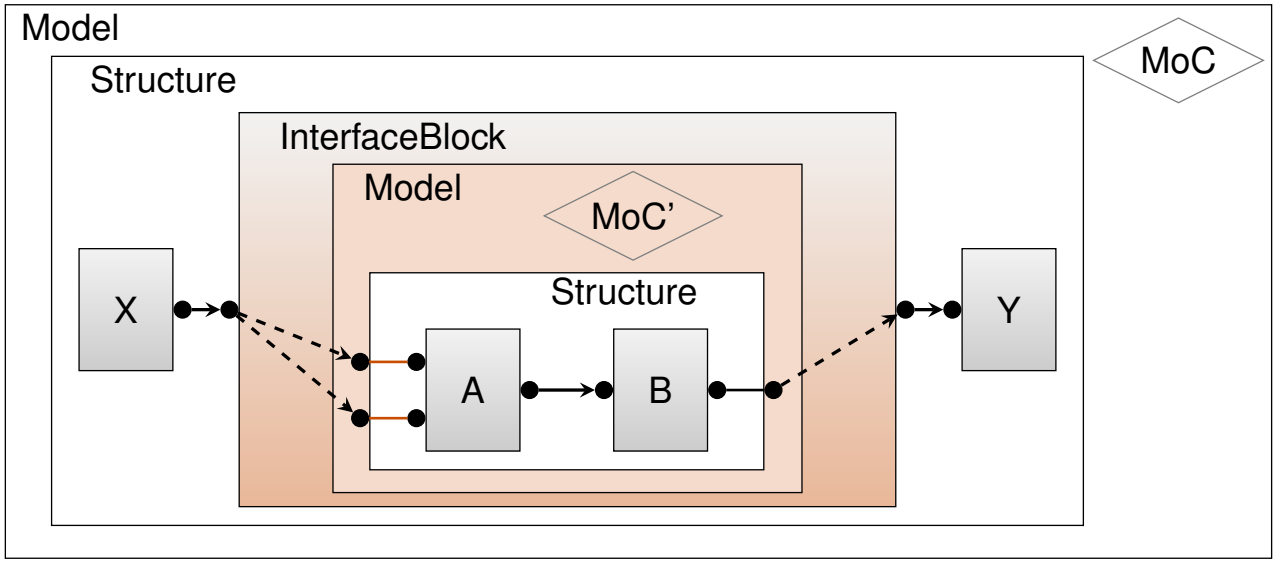
ModHel'X: Generic Execution Engine



The update of an interface block consists in:

- Semantic adaption from MoC to MoC'

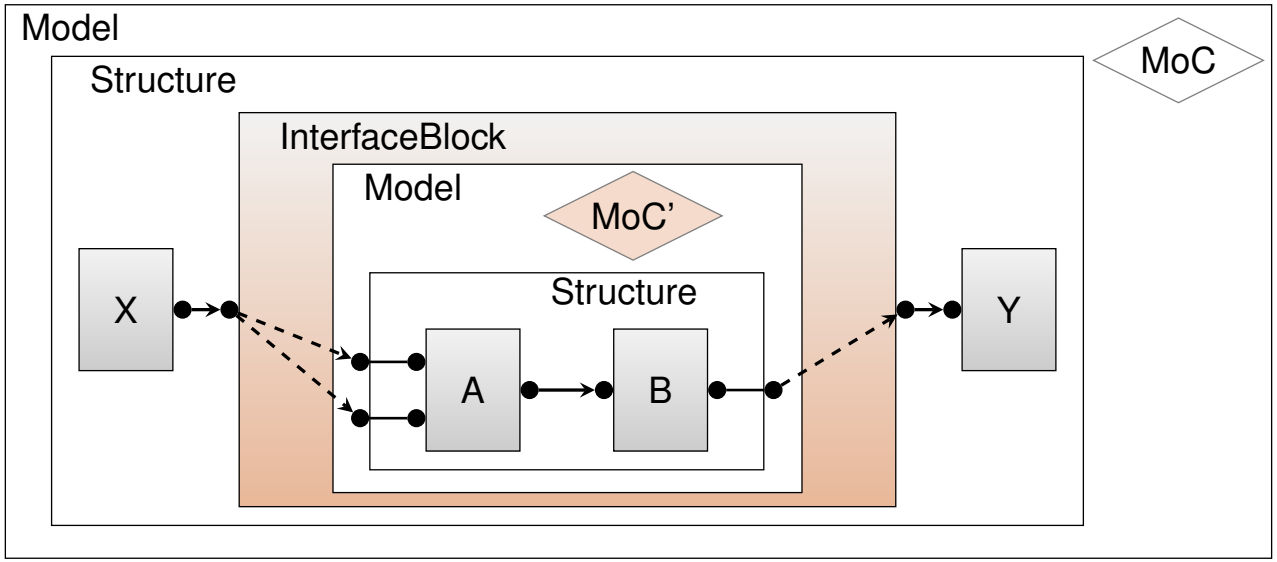
ModHel'X: Generic Execution Engine



The update of an interface block consists in:

- Semantic adaption from MoC to MoC'
- Update of the internal model if there is control for it

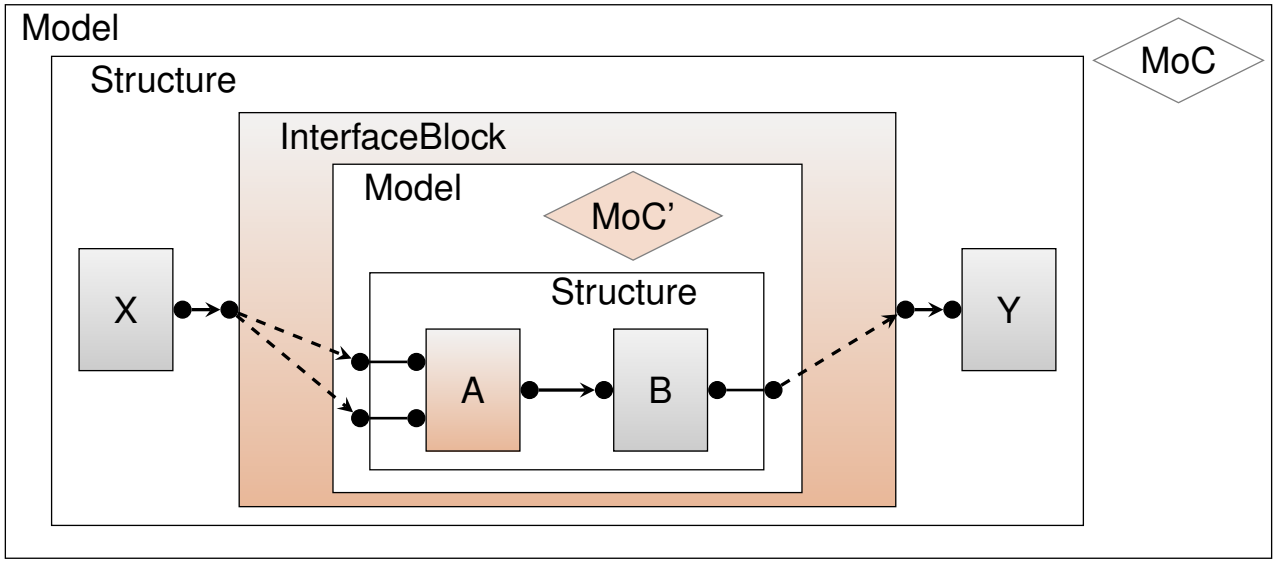
ModHel'X: Generic Execution Engine



The update of an interface block consists in:

- Semantic adaption from MoC to MoC'
- Update of the internal model if there is control for it
- Which gives control to the internal MoC

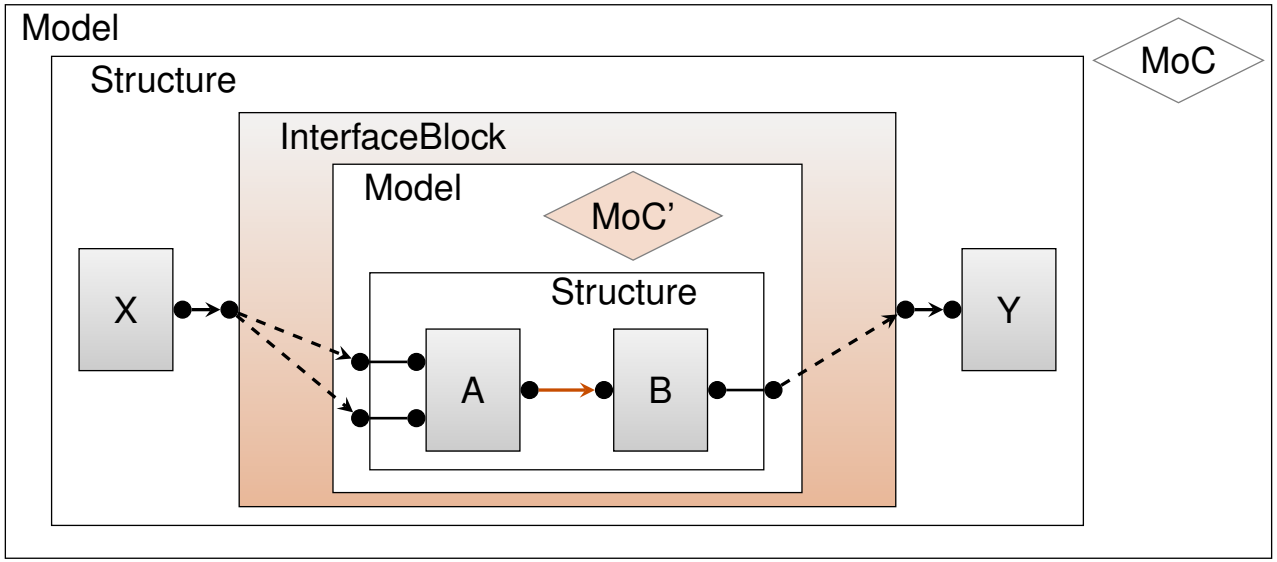
ModHel'X: Generic Execution Engine



The update of an interface block consists in:

- Semantic adaption from MoC to MoC'
- Update of the internal model if there is control for it
- Which gives control to the internal MoC
- Which schedules, updates and propagates

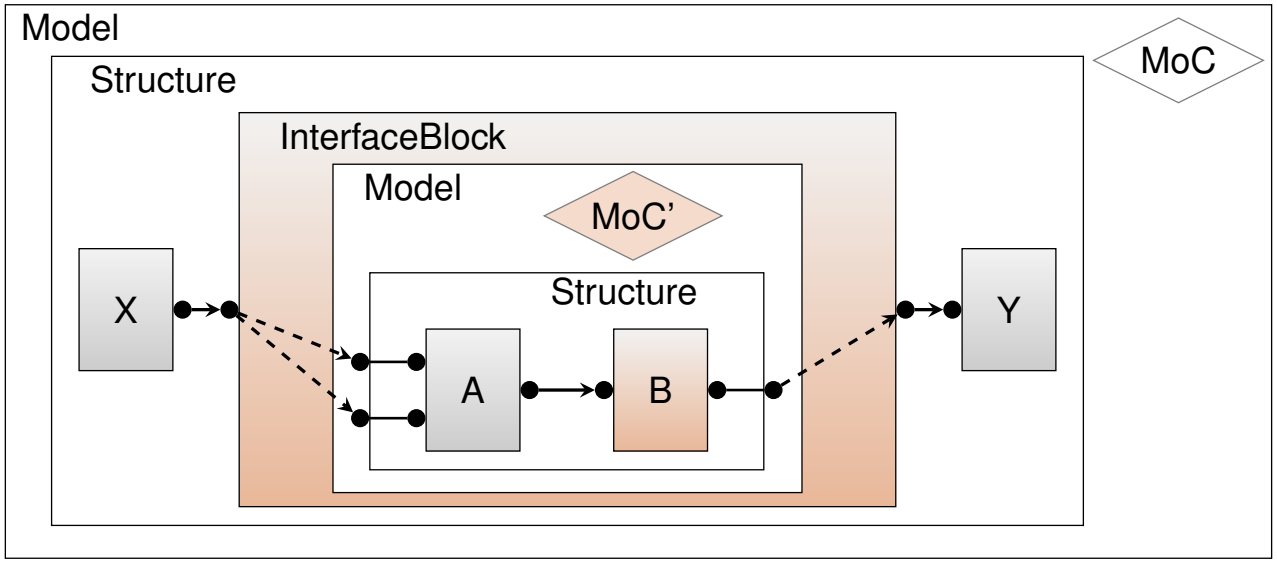
ModHel'X: Generic Execution Engine



The update of an interface block consists in:

- Semantic adaption from MoC to MoC'
- Update of the internal model if there is control for it
- Which gives control to the internal MoC
- Which schedules, updates and propagates

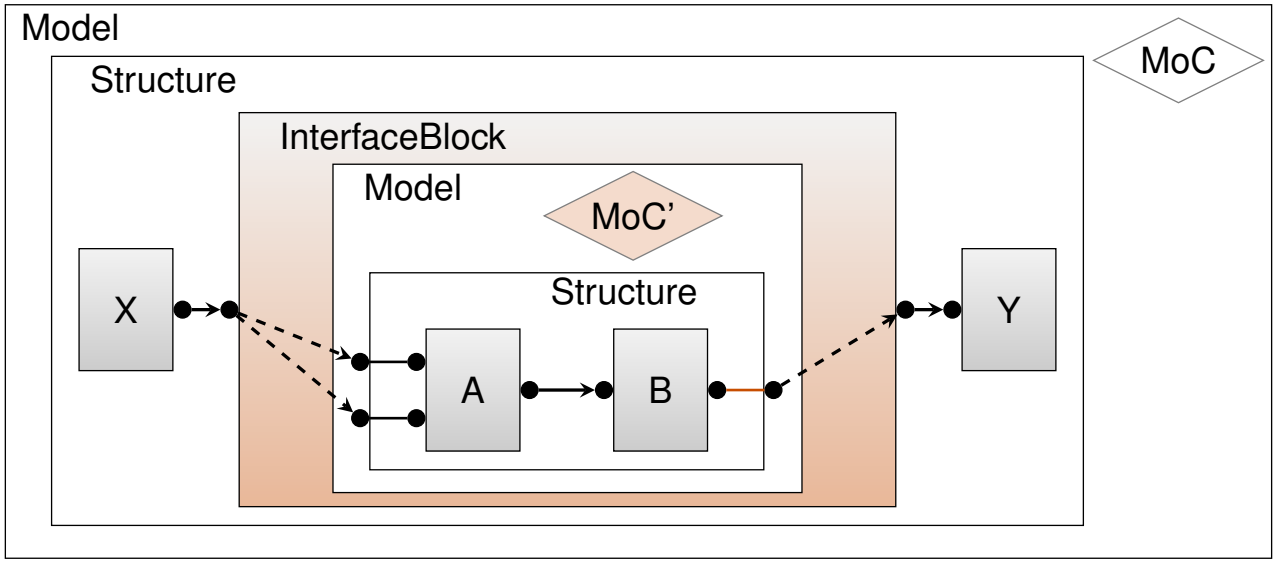
ModHel'X: Generic Execution Engine



The update of an interface block consists in:

- Semantic adaption from MoC to MoC'
- Update of the internal model if there is control for it
- Which gives control to the internal MoC
- Which schedules, updates and propagates

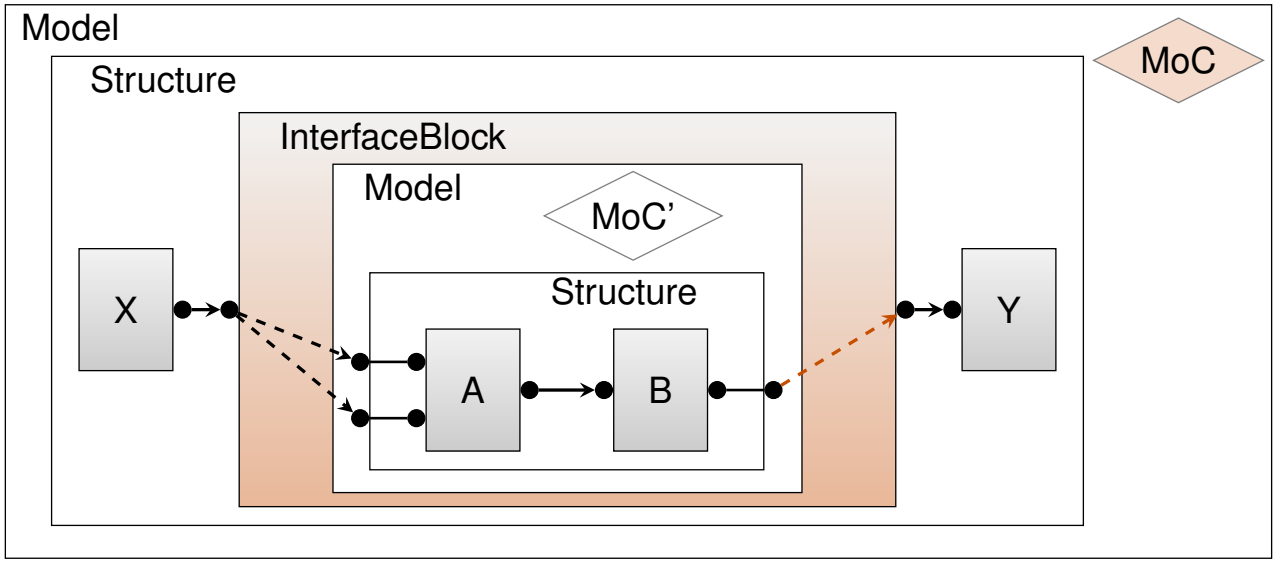
ModHel'X: Generic Execution Engine



The update of an interface block consists in:

- Semantic adaption from MoC to MoC'
- Update of the internal model if there is control for it
- Which gives control to the internal MoC
- Which schedules, updates and propagates
- Transfer of the outputs

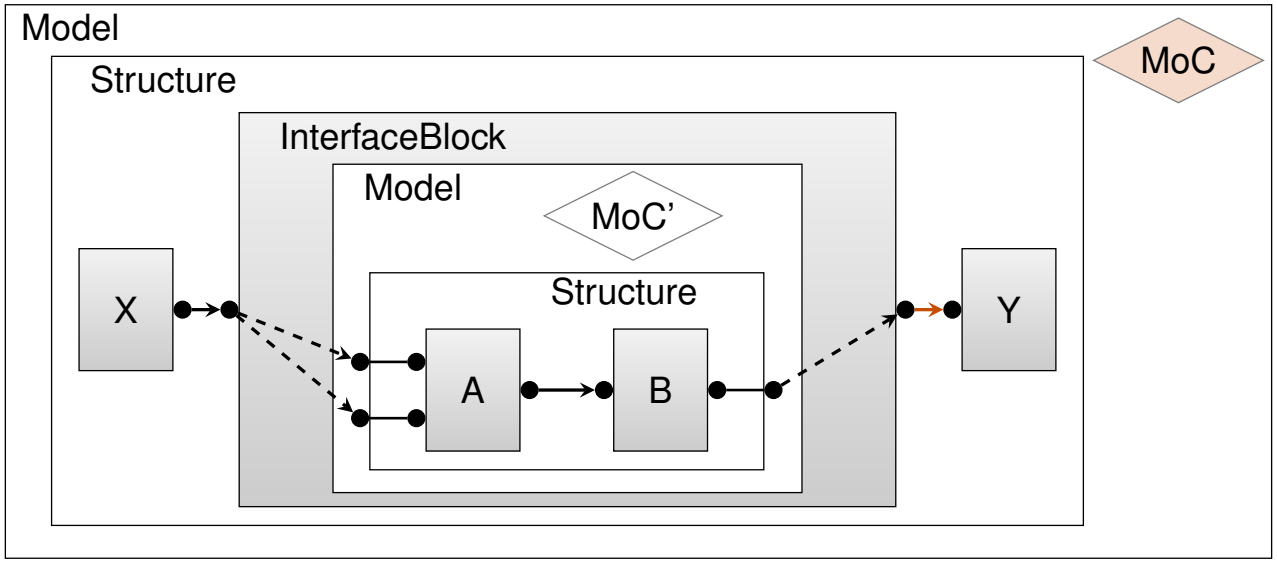
ModHel'X: Generic Execution Engine



The update of an interface block consists in:

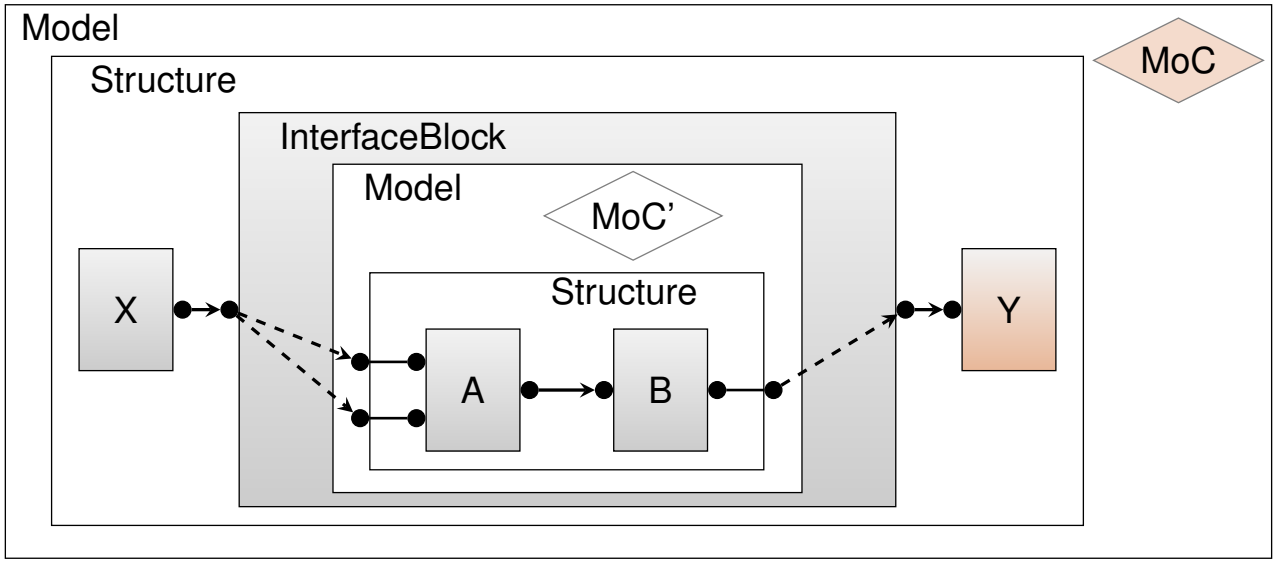
- Semantic adaption from MoC to MoC'
- Update of the internal model if there is control for it
- Which gives control to the internal MoC
- Which schedules, updates and propagates
- Transfer of the outputs
- Semantic adaptation from MoC' to MoC

ModHel'X: Generic Execution Engine



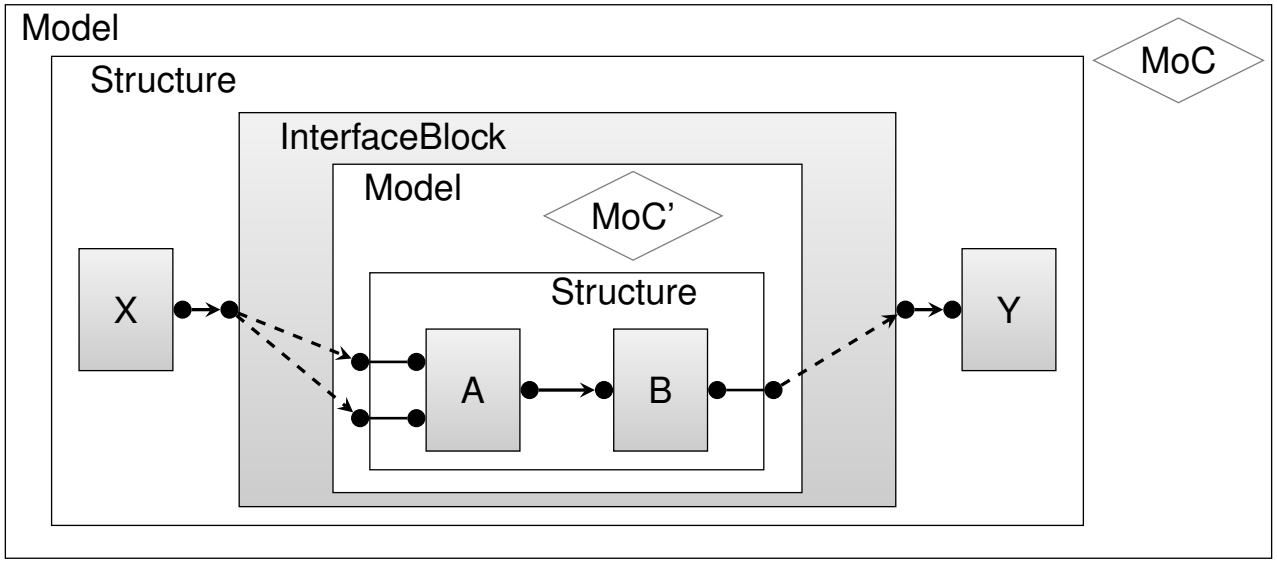
The external MoC propagates the outputs of the IB

ModHel'X: Generic Execution Engine



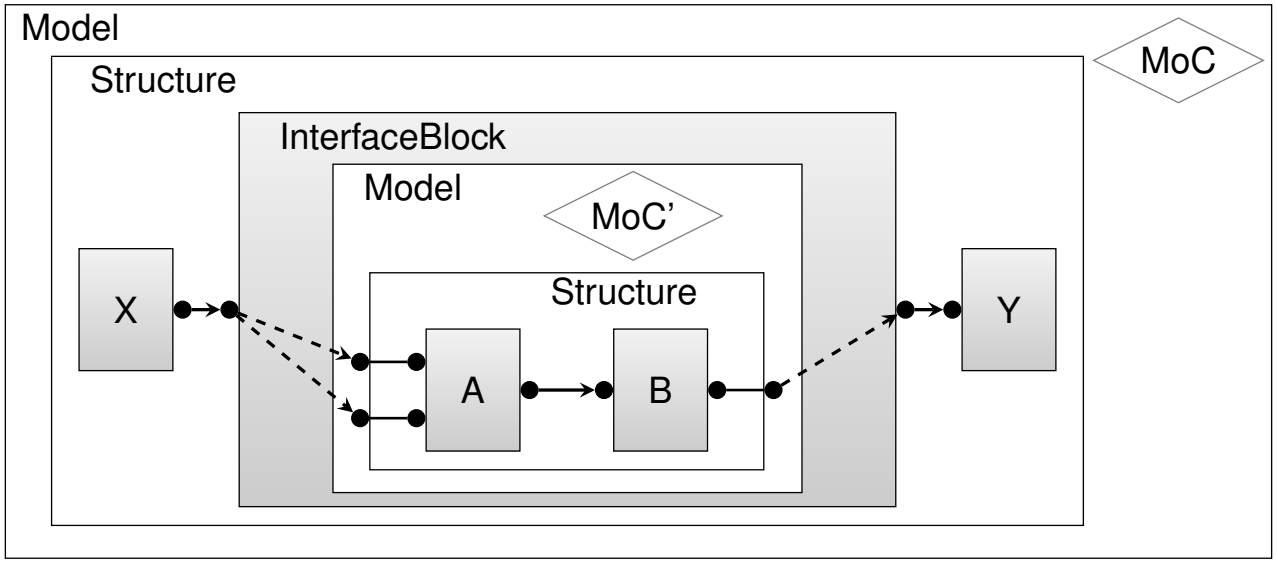
And schedules block Y for update

ModHel'X: Generic Execution Engine



End of the snapshot

ModHel'X: Generic Execution Engine



The real stuff is a bit more complex, but this is enough for today.

- A MoC defines schedule and propagate operations
- The behavior of a block is in its update operation
- Interface blocks delegate their behavior to a model
- The generic engine relies on these operations to execute models

The power window

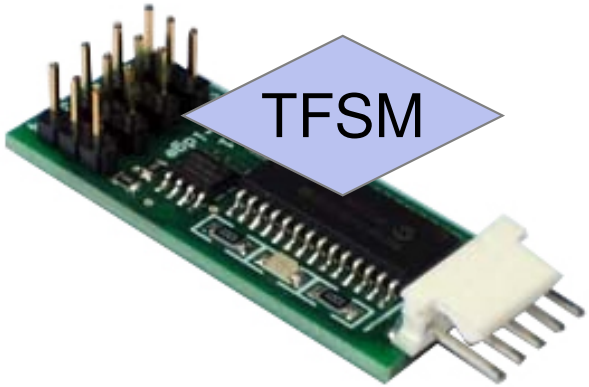
bus

The power window

DE

bus

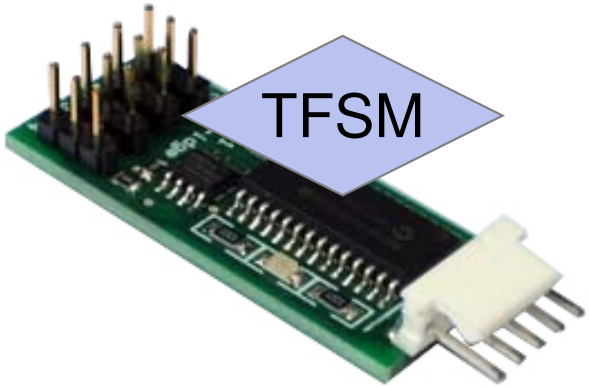
The power window



DE

bus

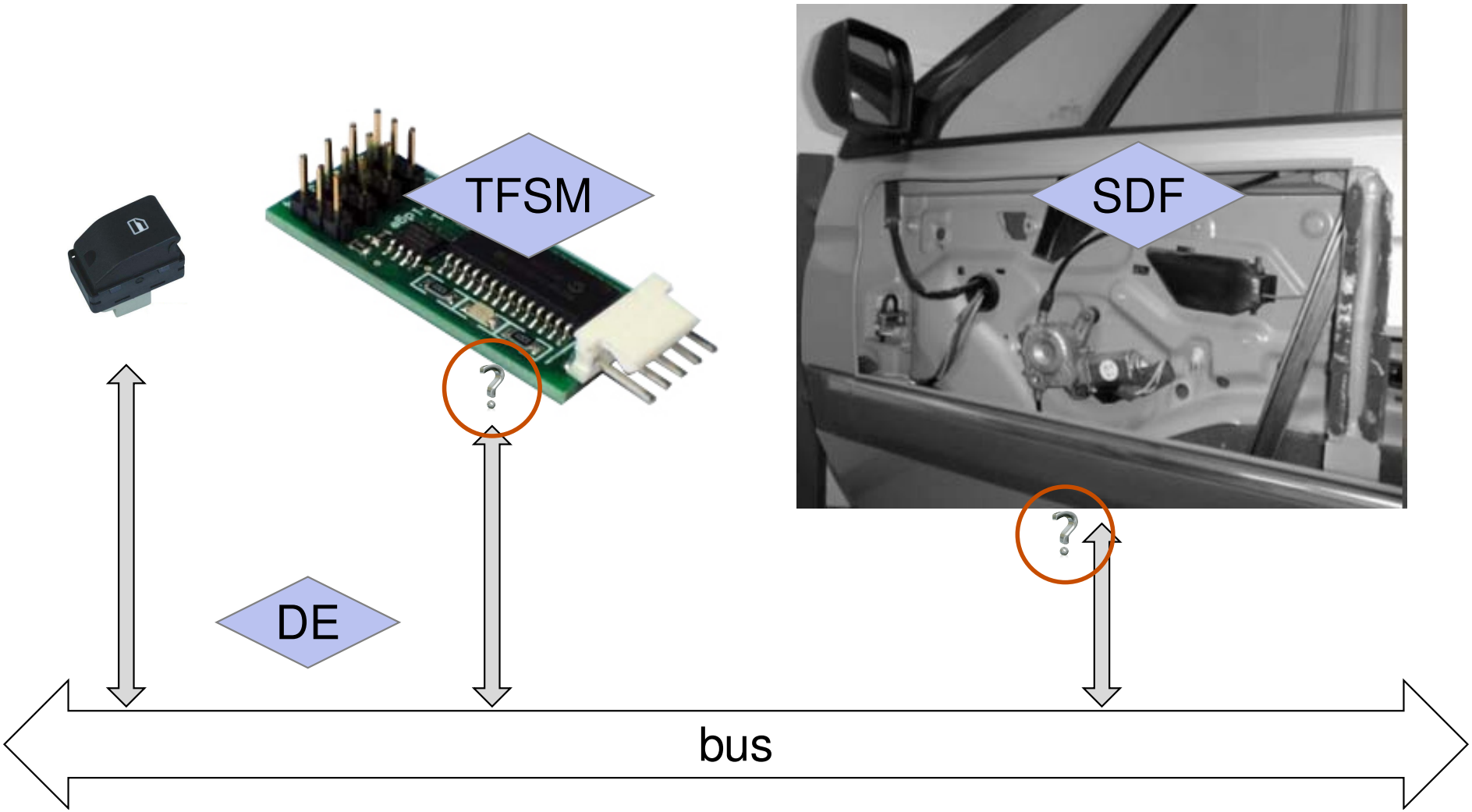
The power window



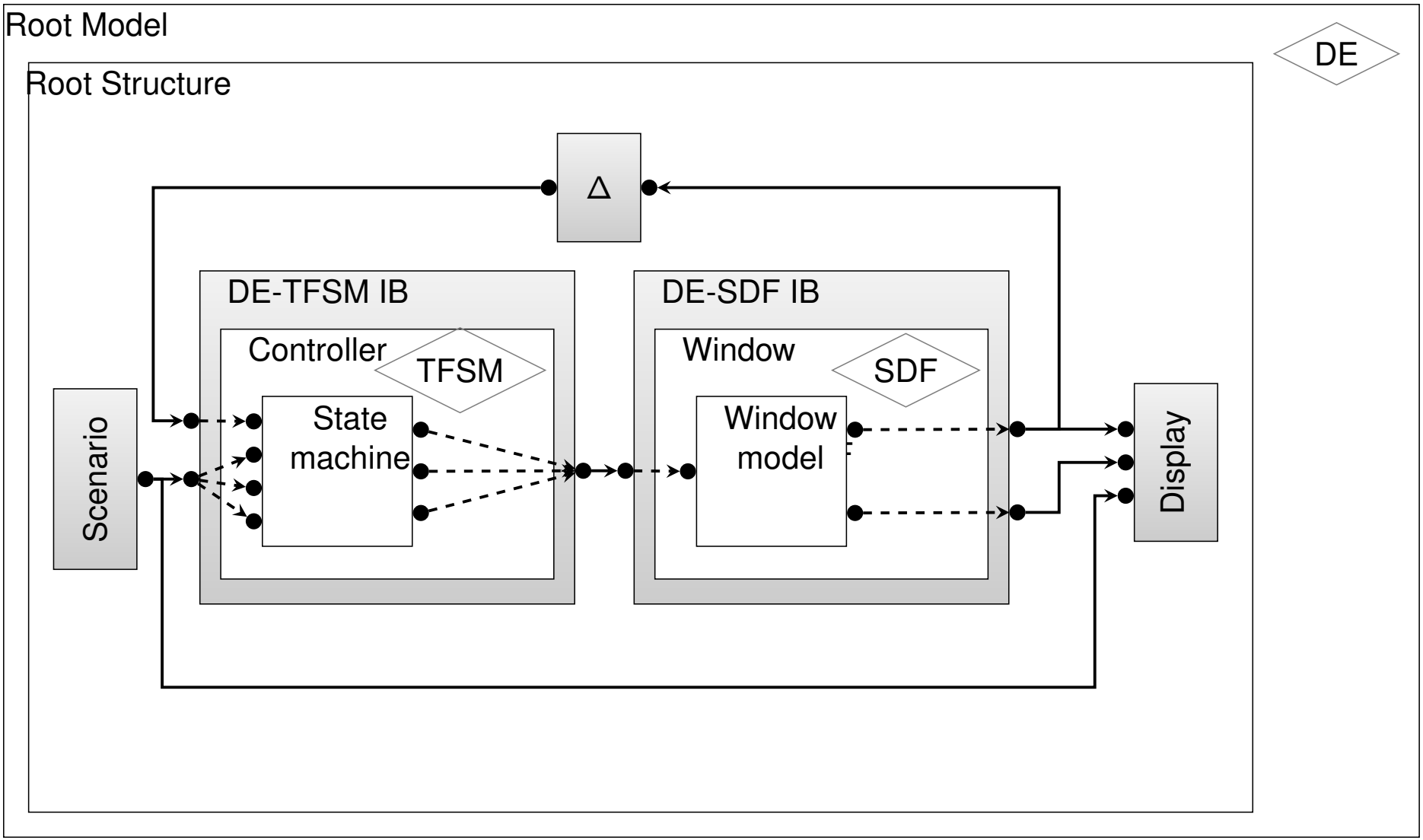
DE

bus

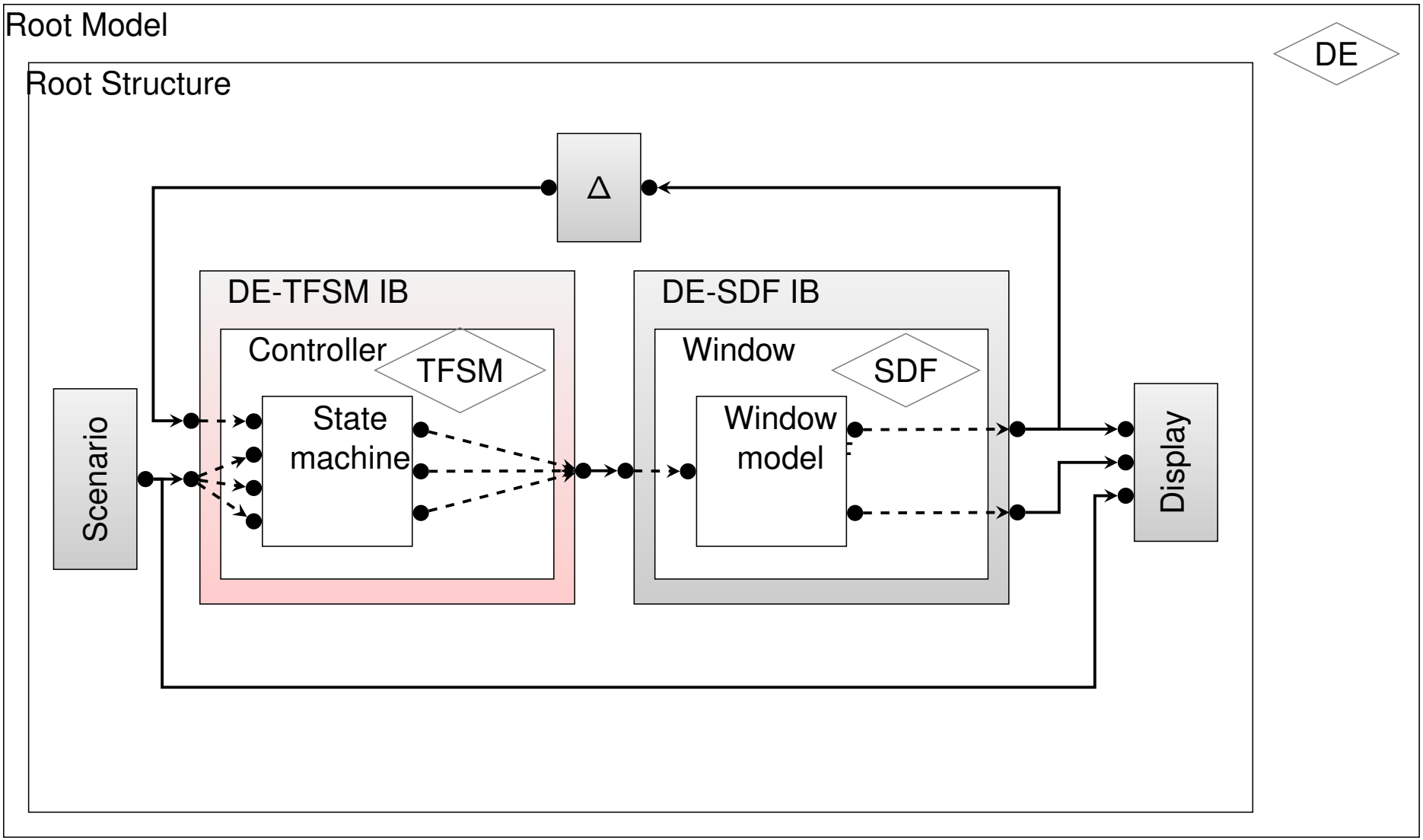
The power window



ModHel'X Model of the Power Window

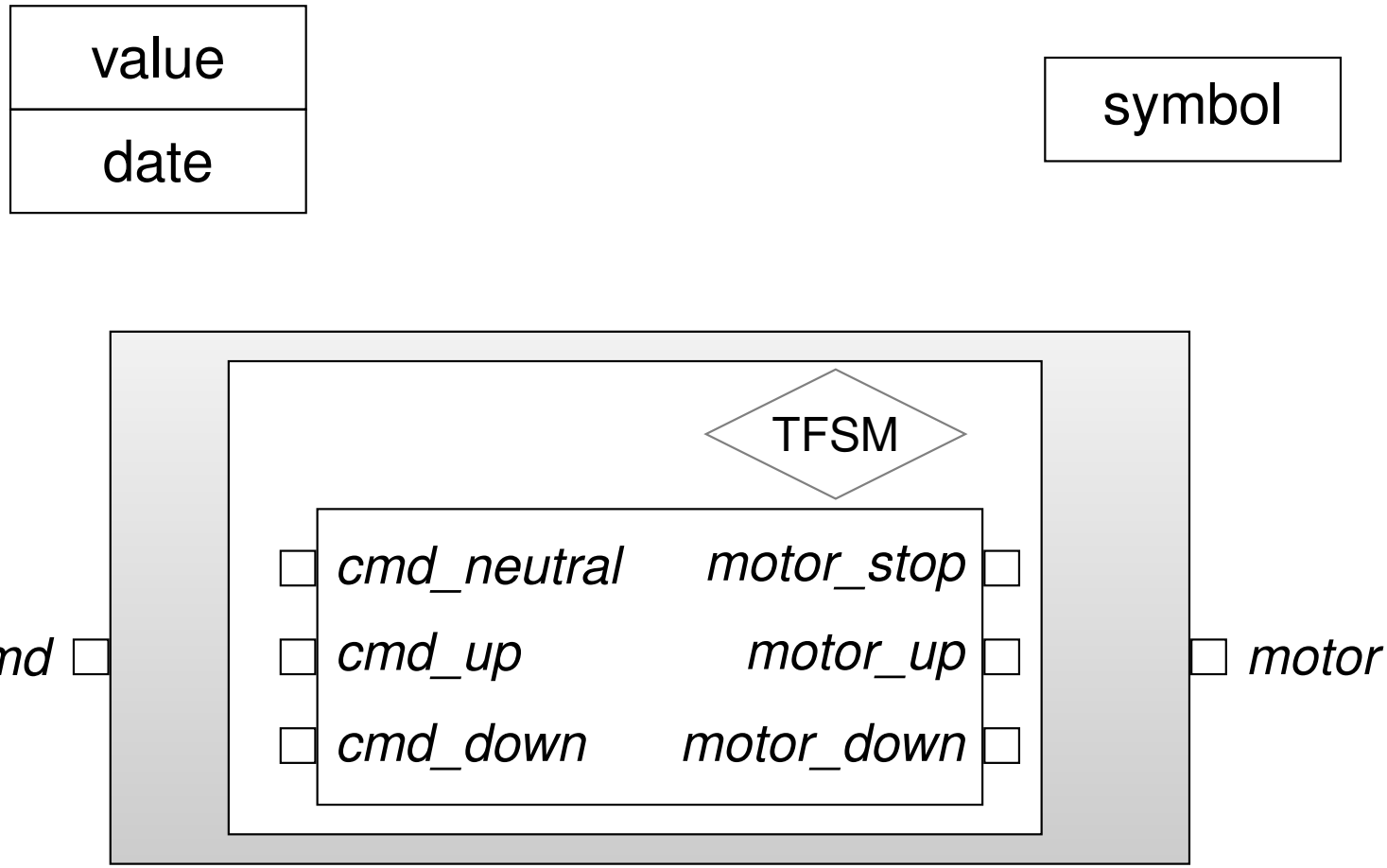


ModHel'X Model of the Power Window



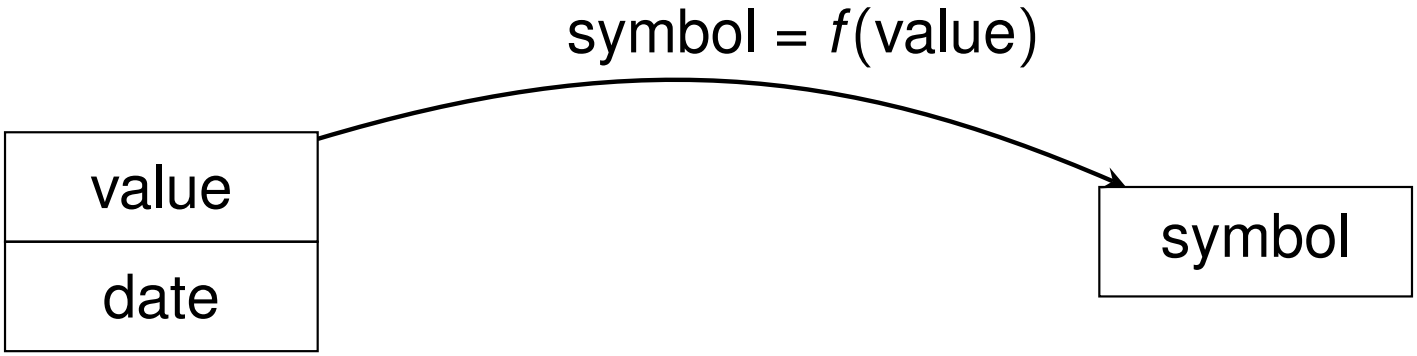
Semantic Adaptation: Data

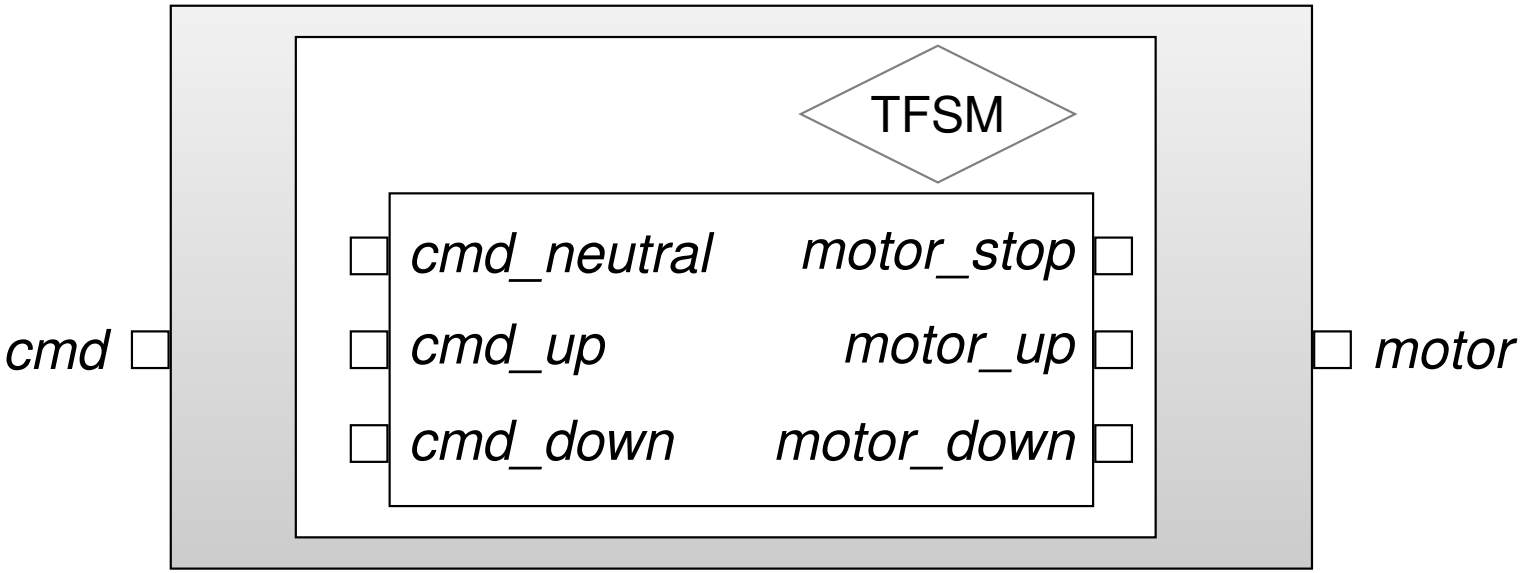
Adaptation of data between DE and TFSM



Semantic Adaptation: Data

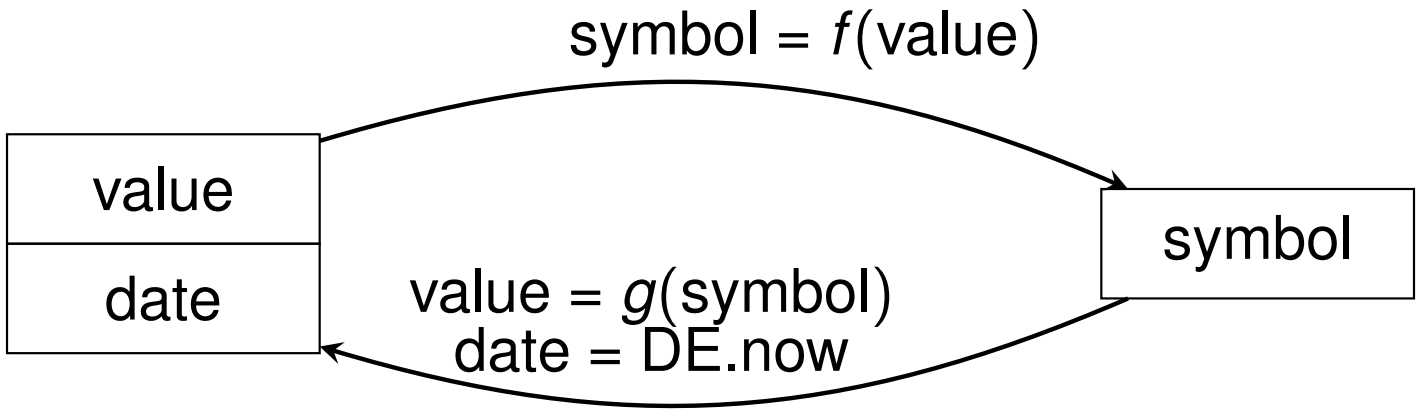
Adaptation of data between DE and TFSM

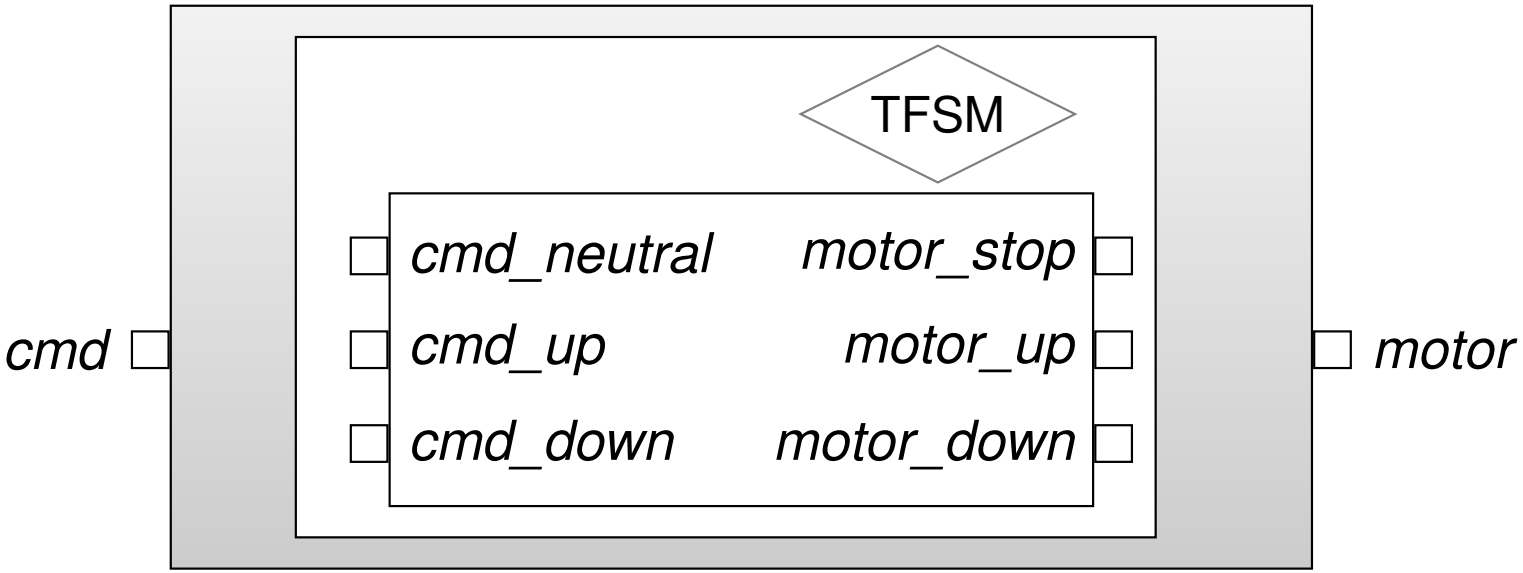




Semantic Adaptation: Data

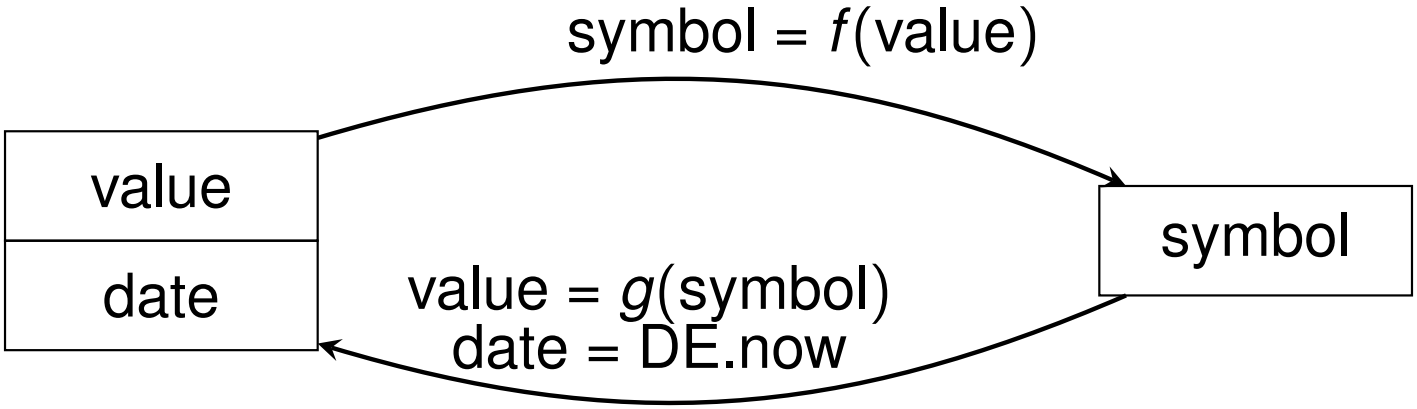
Adaptation of data between DE and TFSM

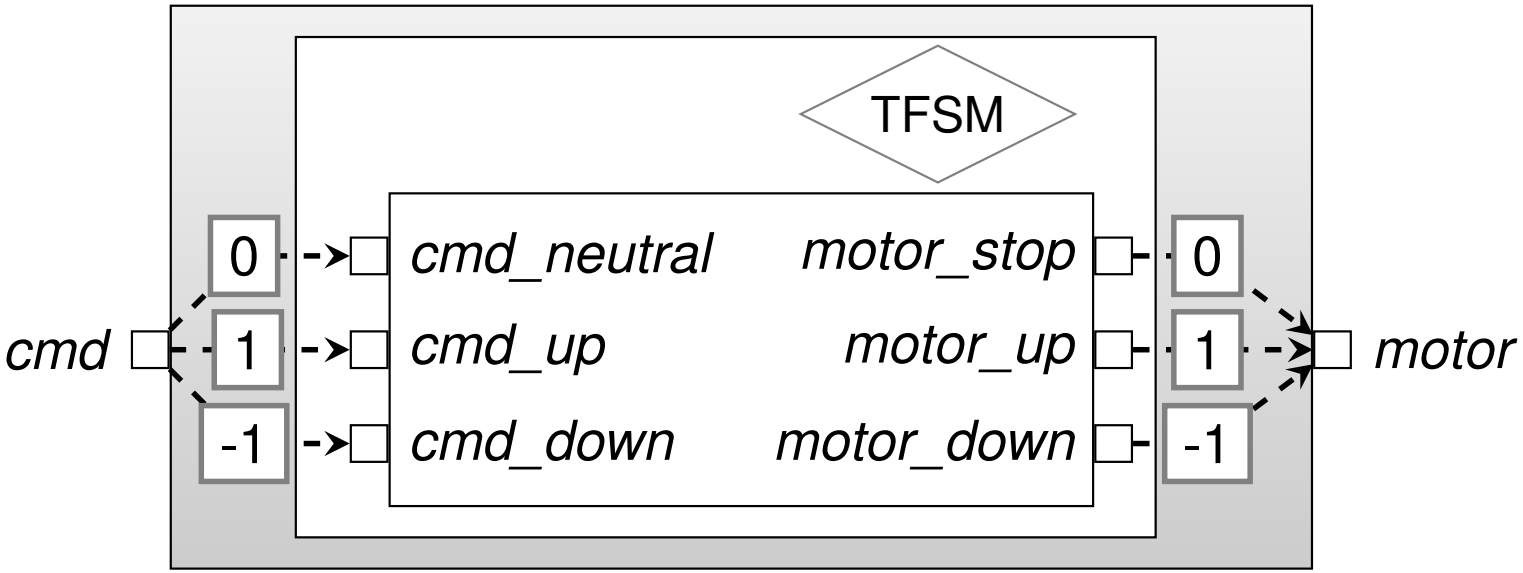




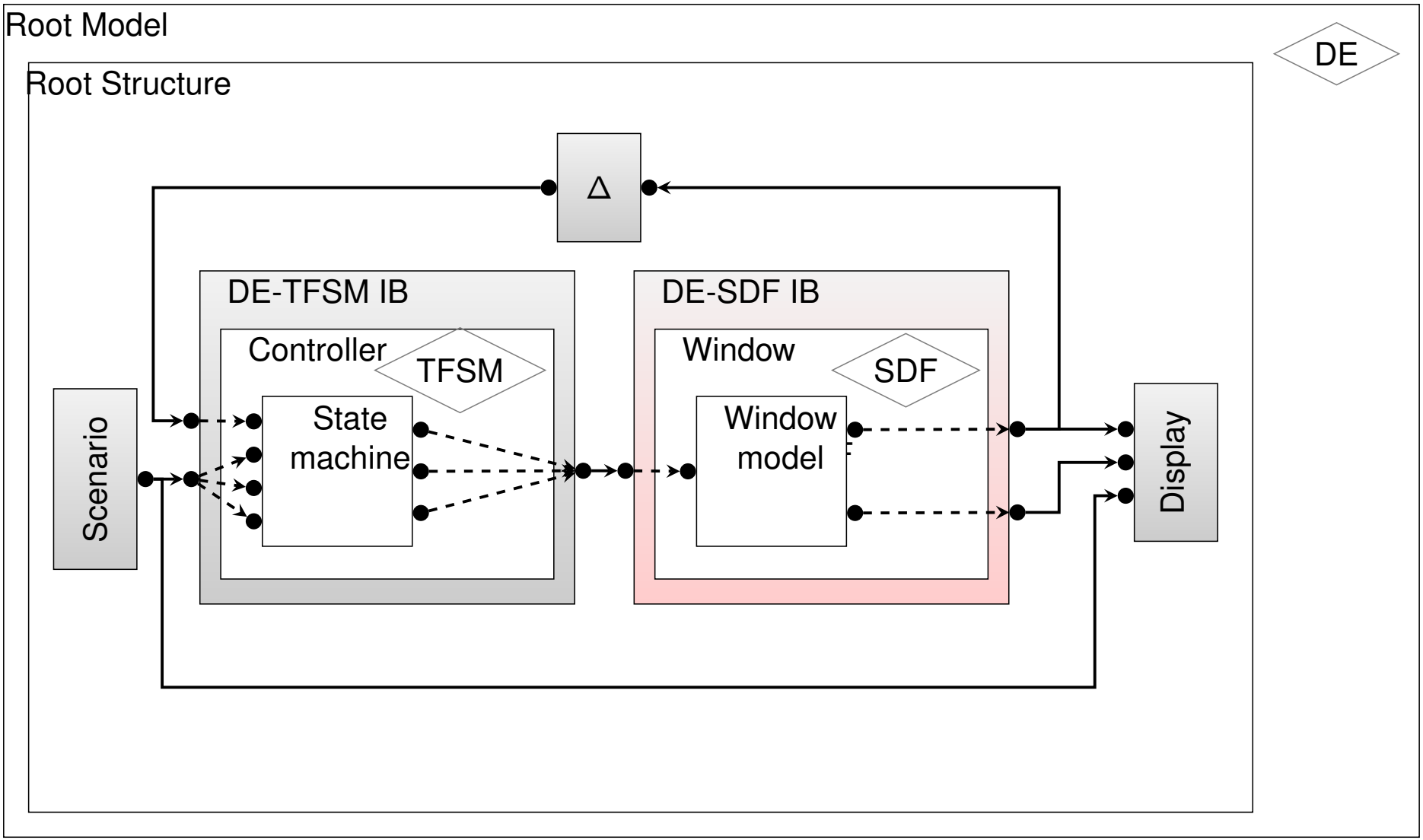
Semantic Adaptation: Data

Adaptation of data between DE and TFSM



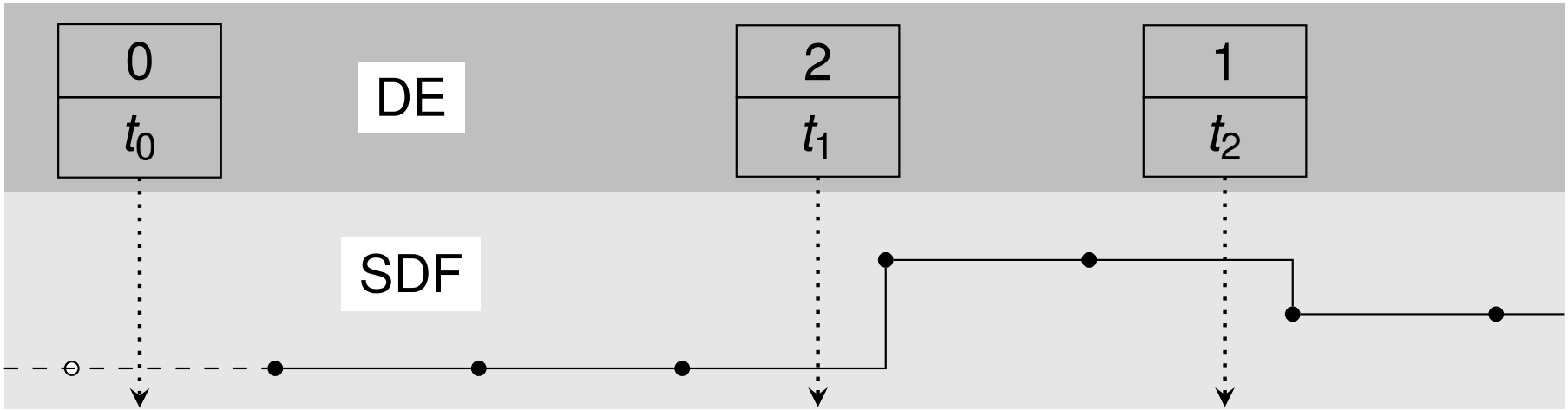


ModHel'X Model of the Power Window



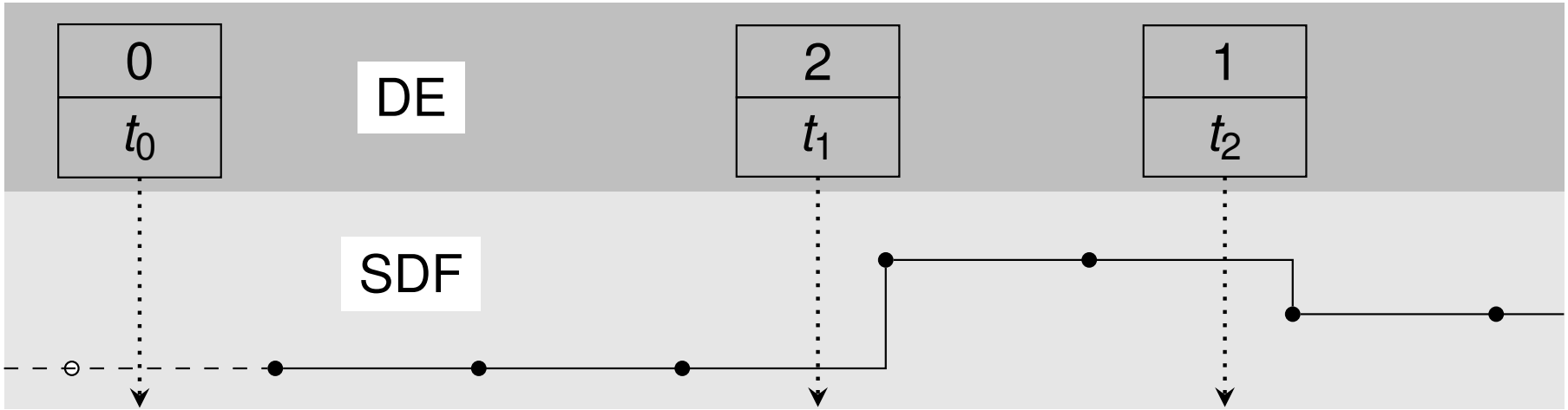
Semantic Adaptation: Data

Adaptation of data between DE and SDF



Semantic Adaptation: Data

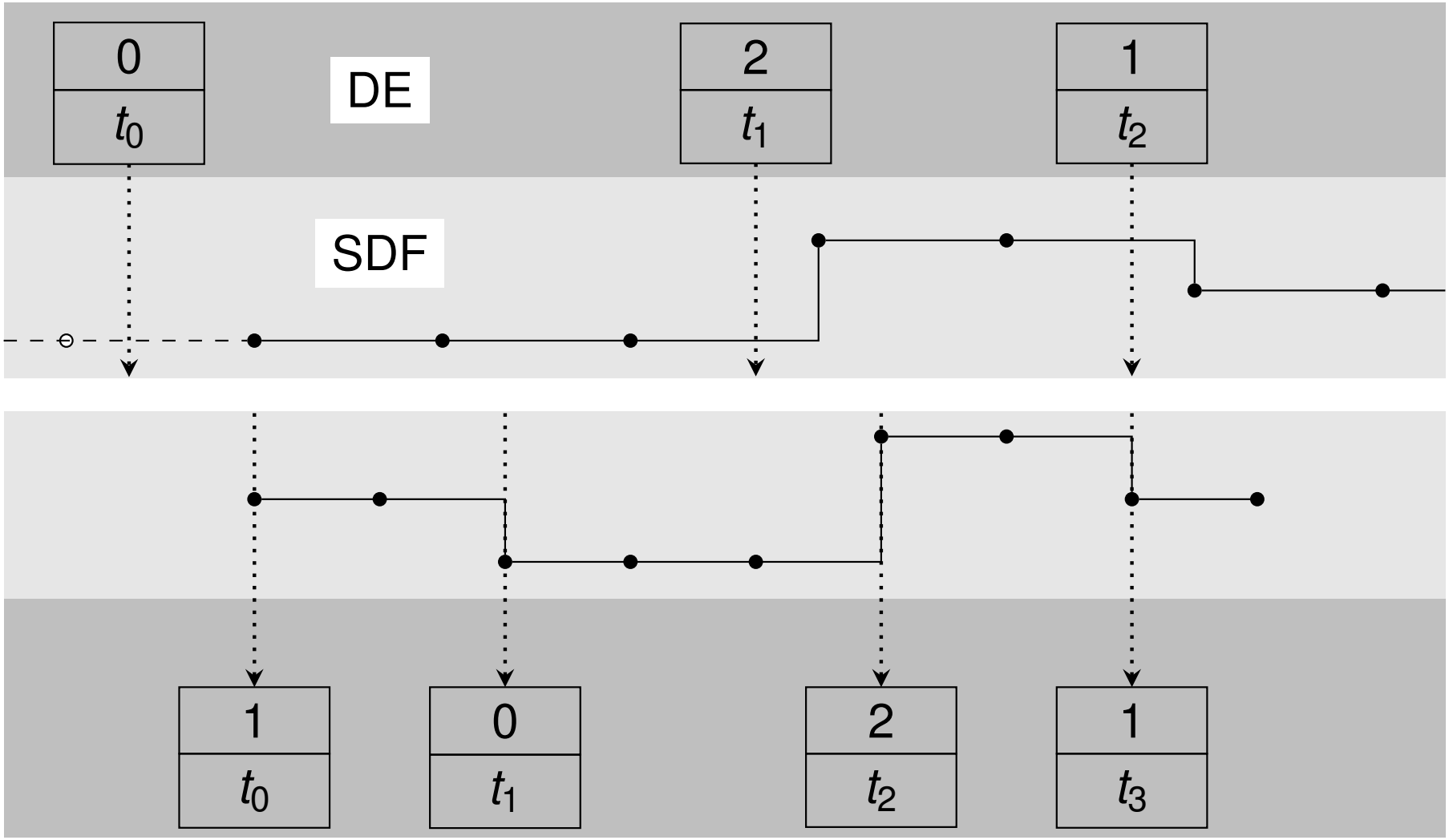
Adaptation of data between DE and SDF



- Relative positioning of events / samples $\Rightarrow$ adaptation of time
- Occurrence of samples $\Rightarrow$ adaptation of control

Semantic Adaptation: Data

Adaptation of data between DE and SDF



Semantic Adaptation: Time and Control

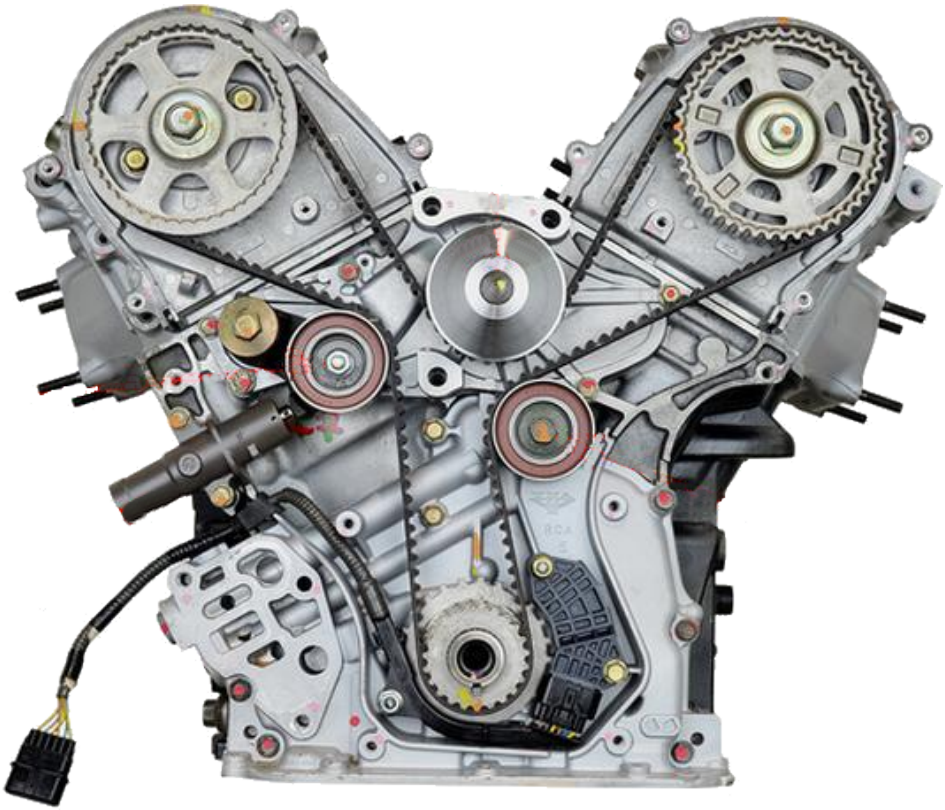
Time

- Time may pass at different paces
- Time units may be of different natures

Semantic Adaptation: Time and Control

Time

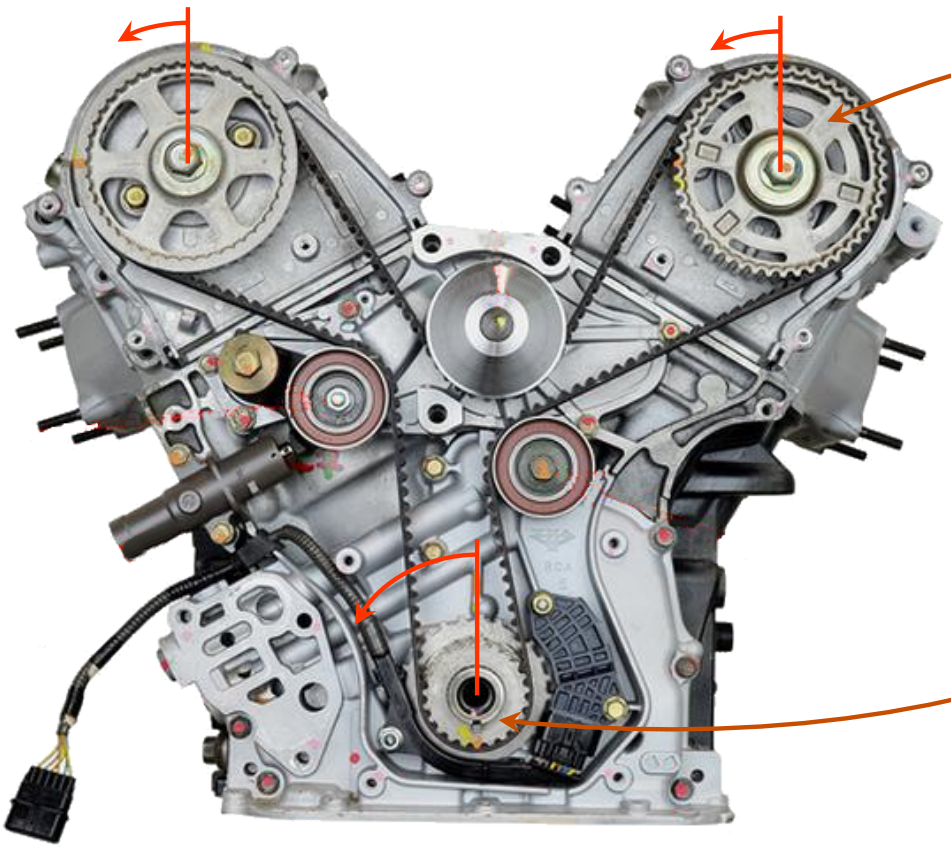
- Time may pass at different paces
- Time units may be of different natures



Semantic Adaptation: Time and Control

Time

- Time may pass at different paces
- Time units may be of different natures

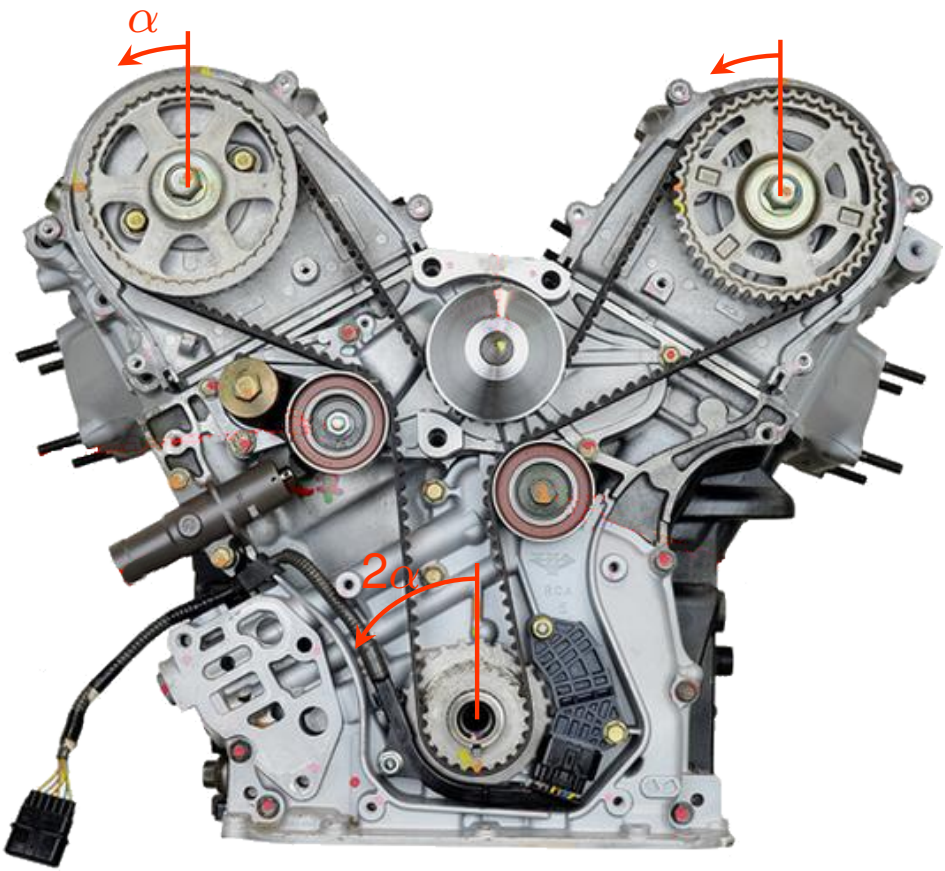


Angular info on camshaft / crankshaft
“Time” is measured in degrees

Semantic Adaptation: Time and Control

Time

- Time may pass at different paces
- Time units may be of different natures



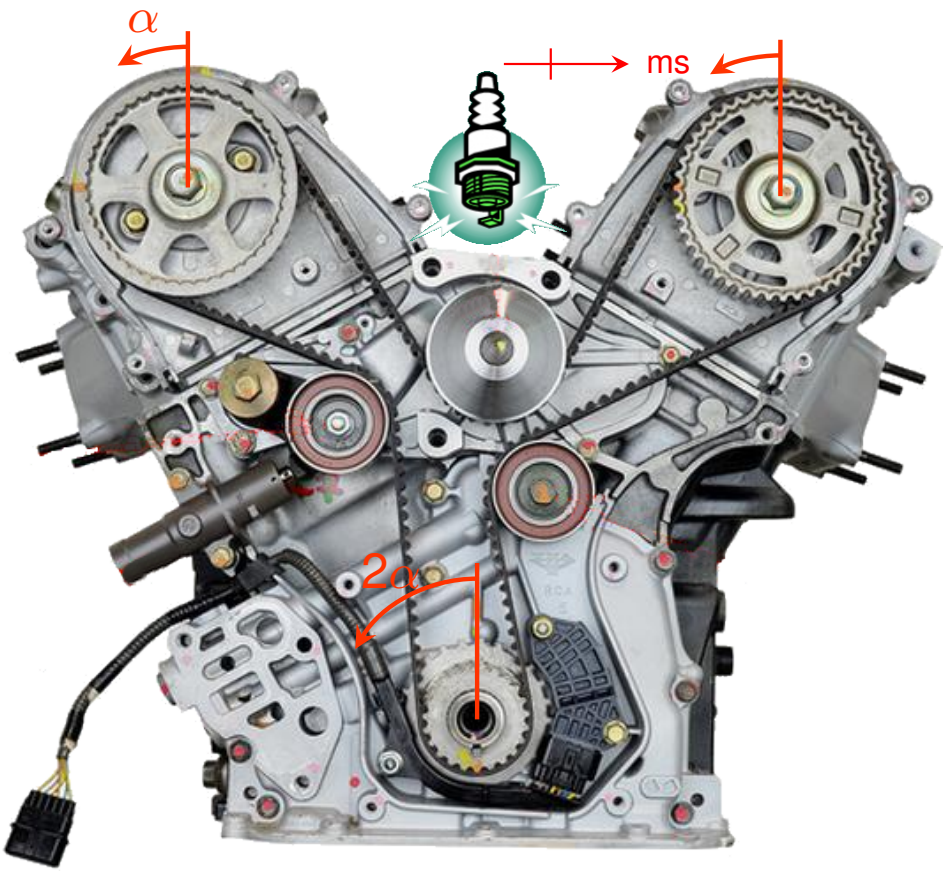
Angular info on camshaft / crankshaft
“Time” is measured in degrees

Time runs twice faster on crankshaft

Semantic Adaptation: Time and Control

Time

- Time may pass at different paces
- Time units may be of different natures



Angular info on camshaft / crankshaft
"Time" is measured in degrees

Time runs twice faster on crankshaft

Time is seconds in ignition system

Seconds/angle depends on engine rpms

Control depends on ignition time

Semantic Adaptation: Time and Control

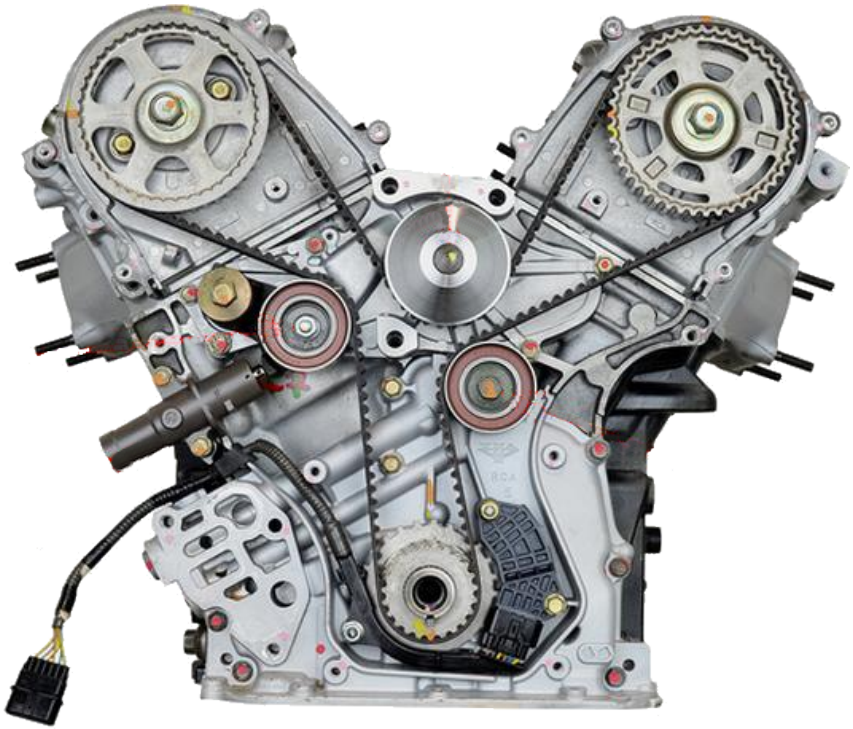
Control

- Control describes when things happen
- Event driven: when something happens
- Time driven: when some time has elapsed

Semantic Adaptation: Time and Control

Control

- Control describes when things happen
- Event driven: when something happens
- Time driven: when some time has elapsed



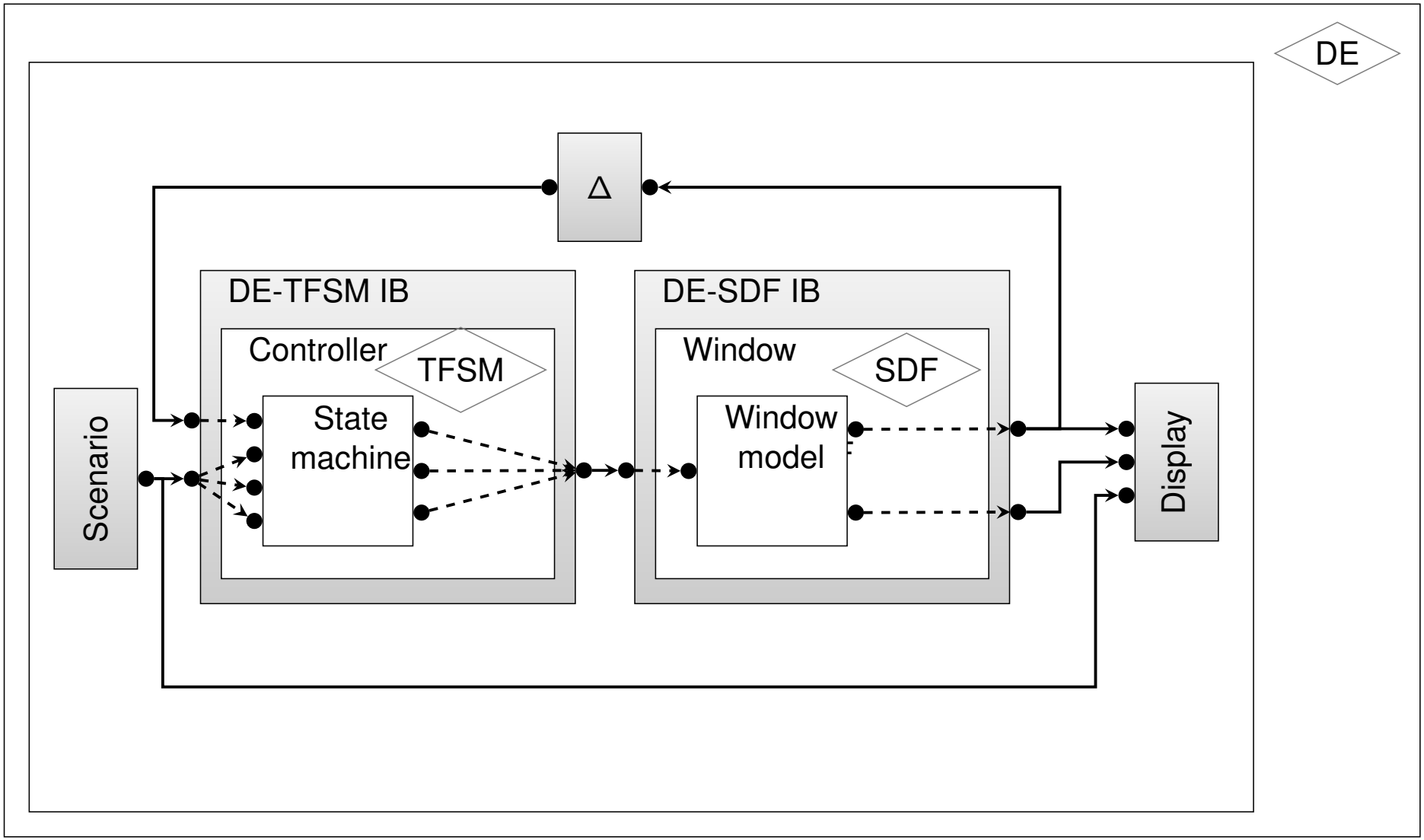
Control depends on ignition time

Ignition is ΔT before Top Dead Center

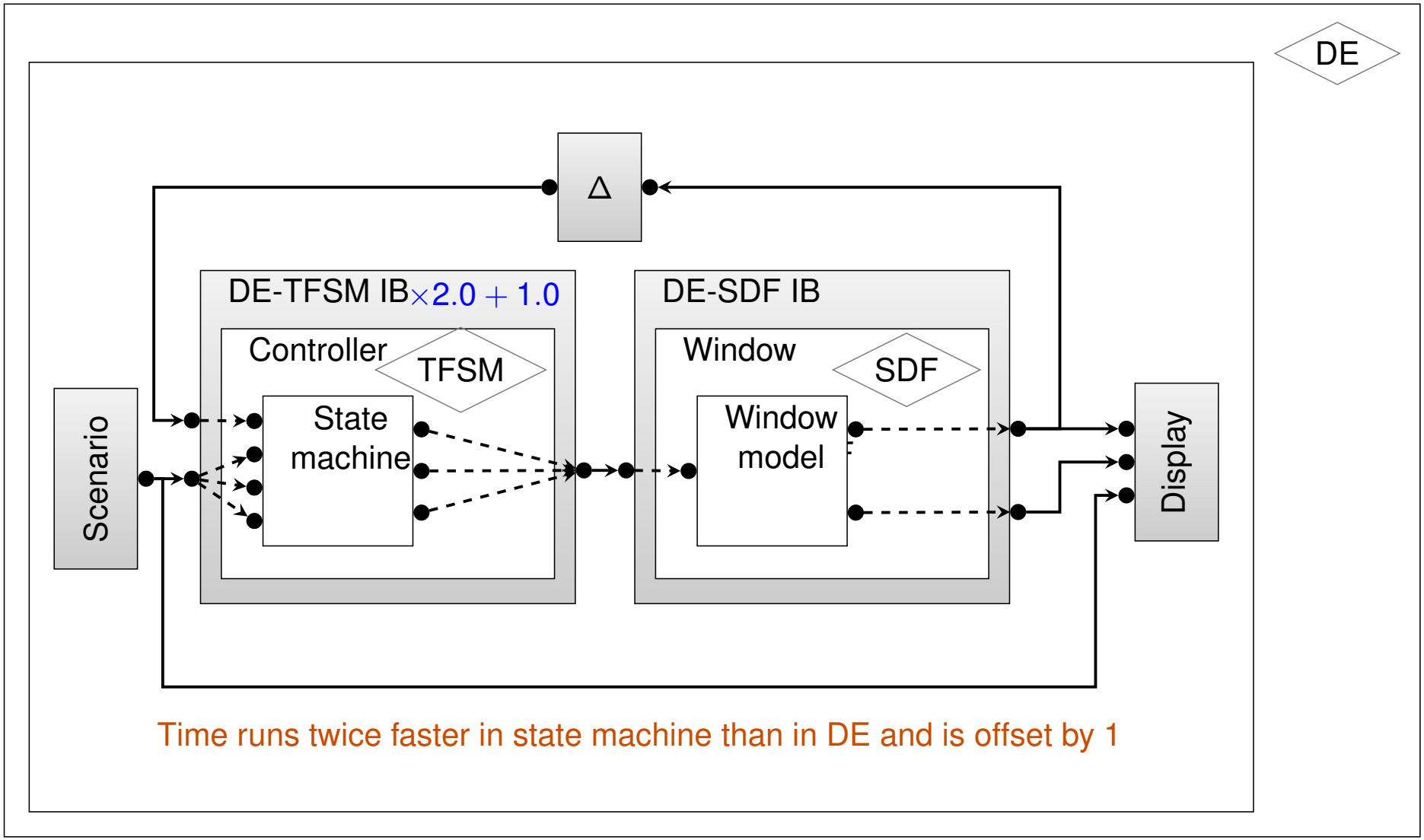
ΔT is at $\alpha°$ on crankshaft

ΔT is at $\frac{\alpha}{2}°$ on camshaft

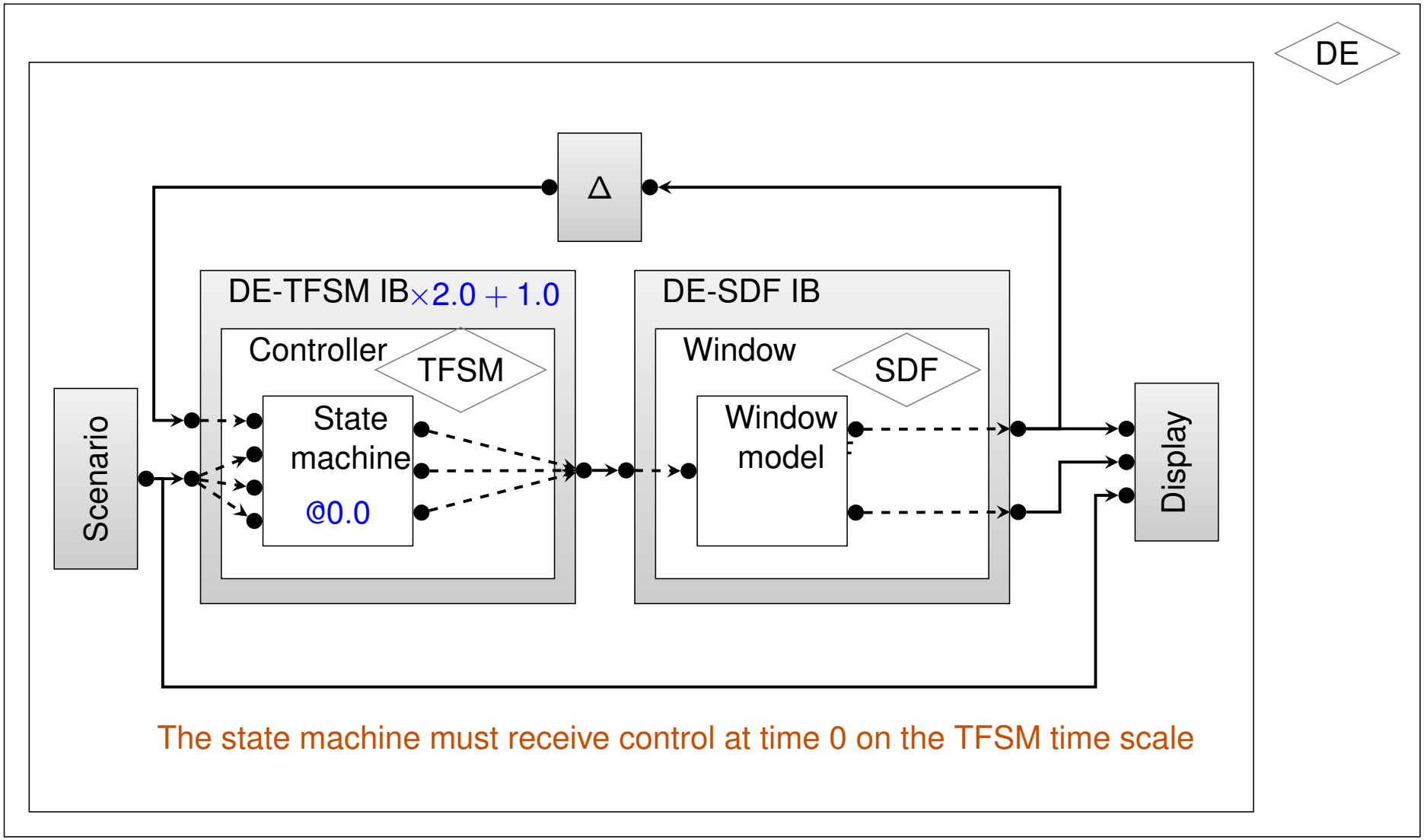
Time and Control in the Power Window



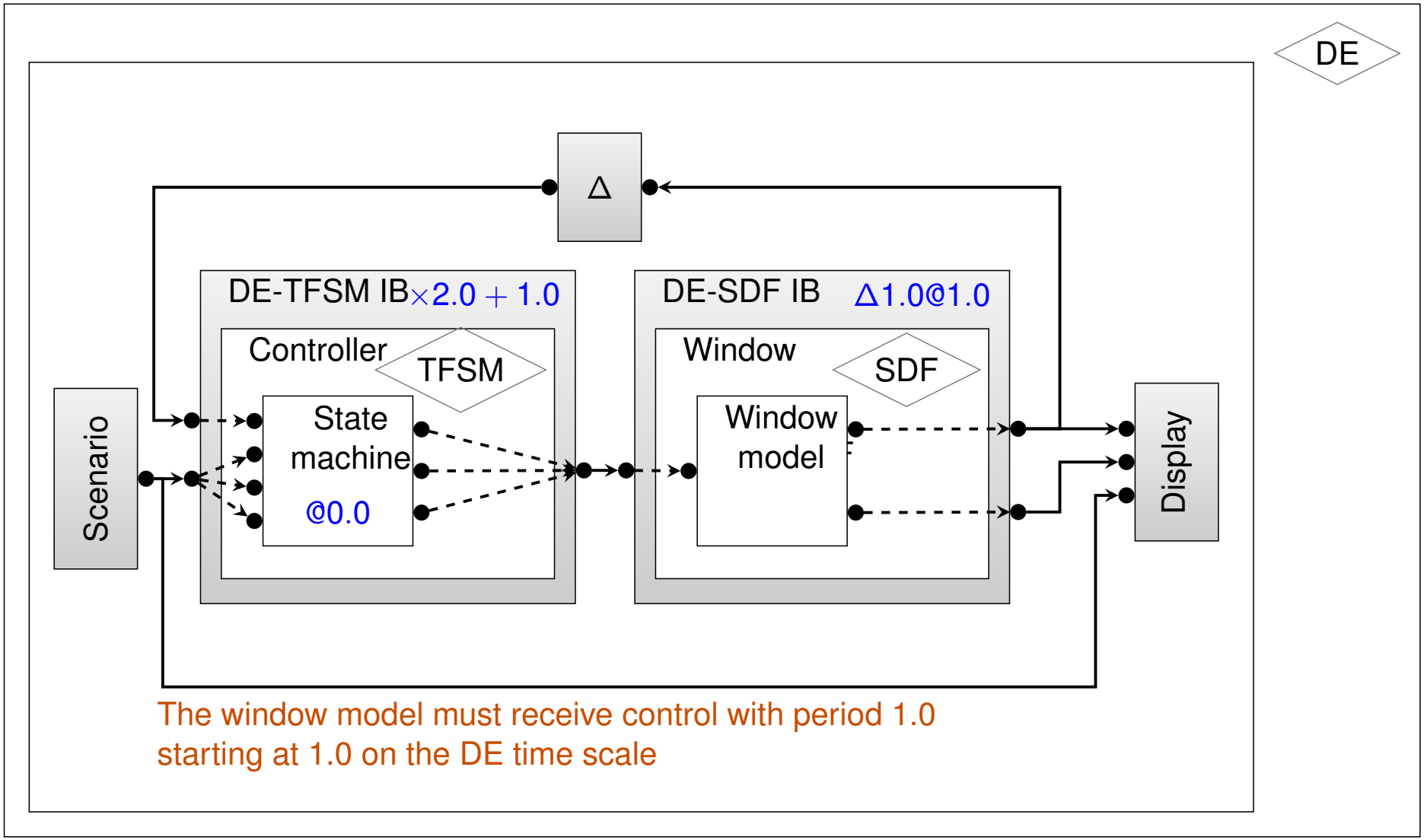
Time and Control in the Power Window



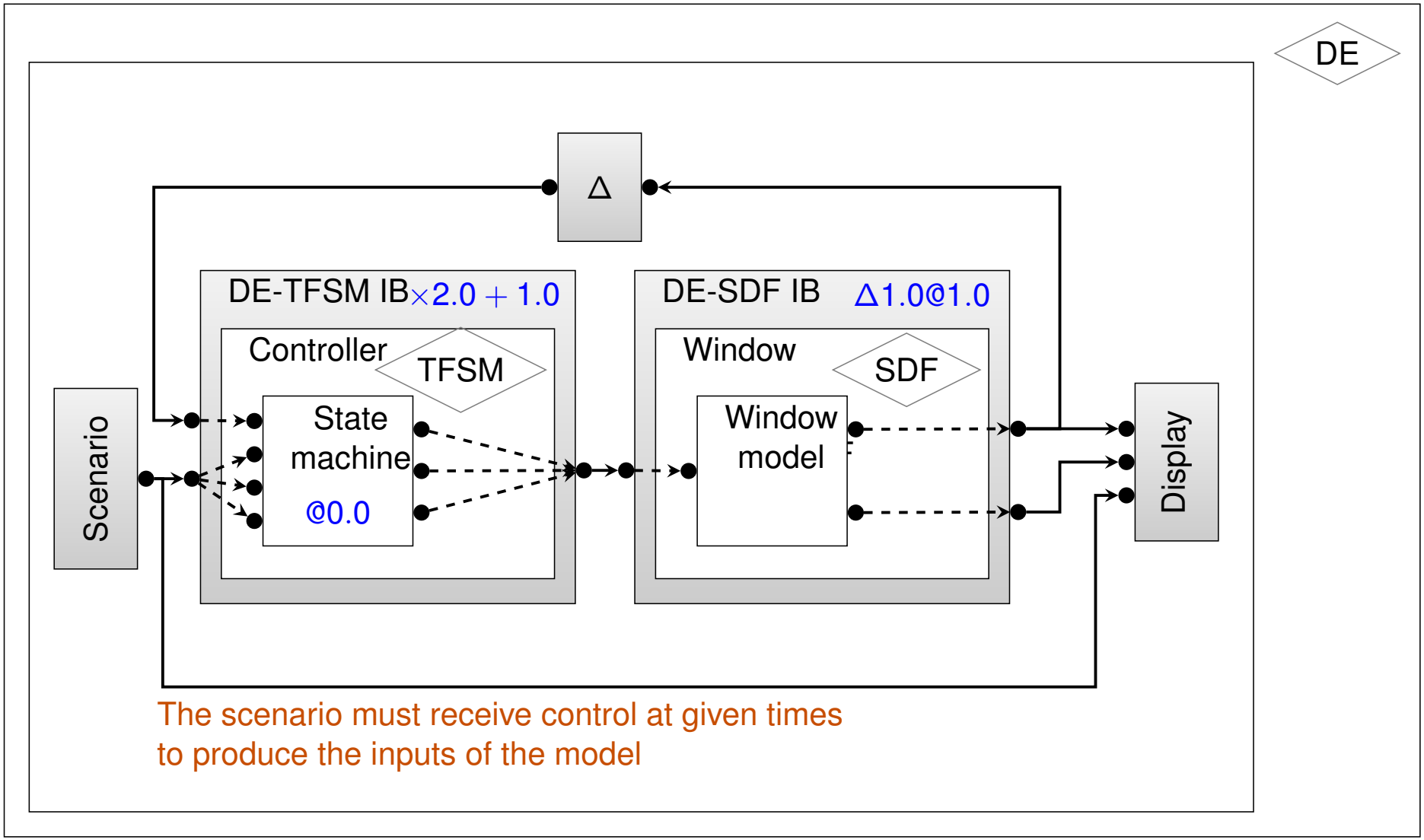
Time and Control in the Power Window



Time and Control in the Power Window

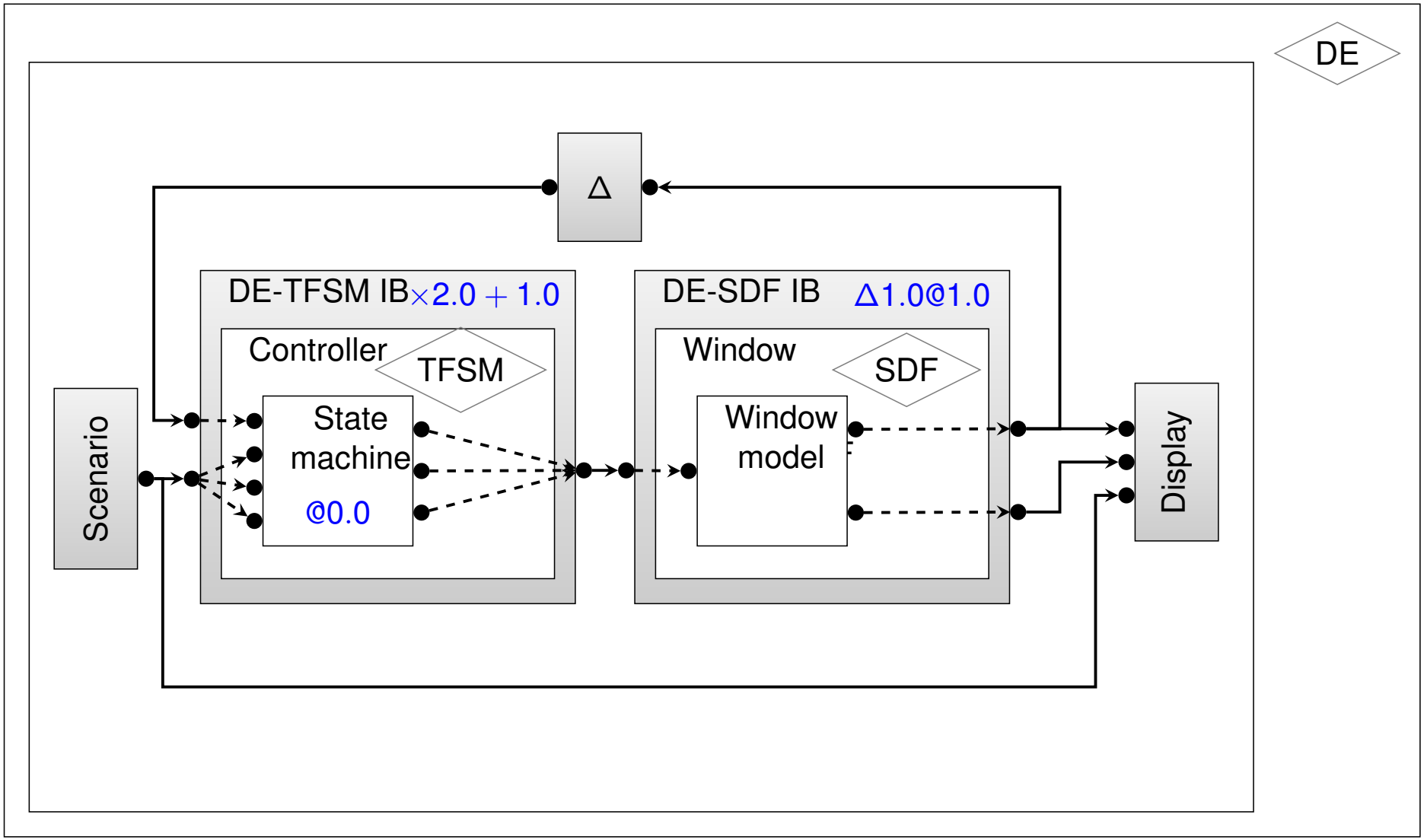


Time and Control in the Power Window



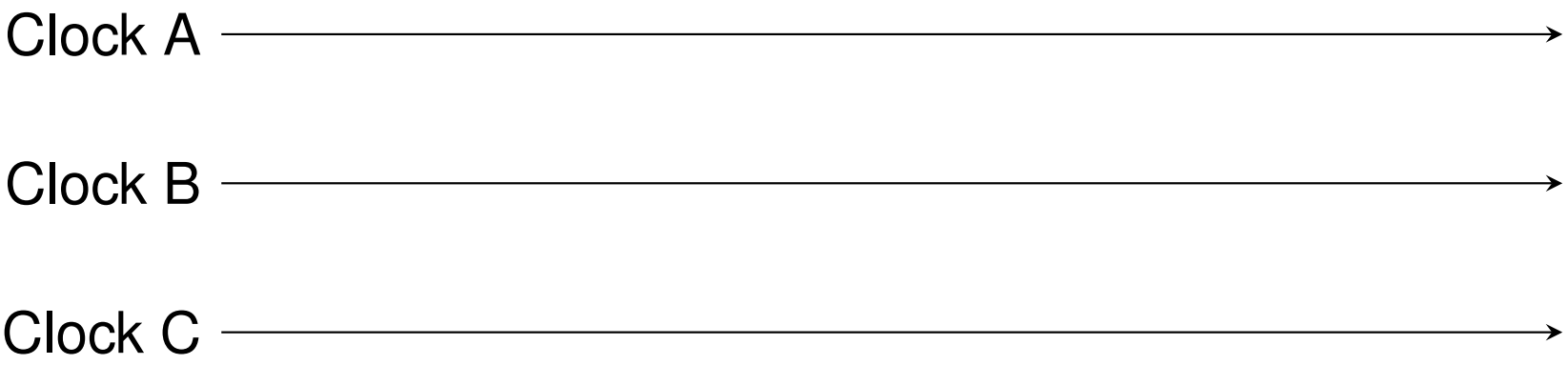
scenario: 1 @2.0; 0 @2.8; 1 @3.0; 0 @5.0; 1 @7.0; 0 @7.4; -1 @10.0; 0 @10.1;

Time and Control in the Power Window



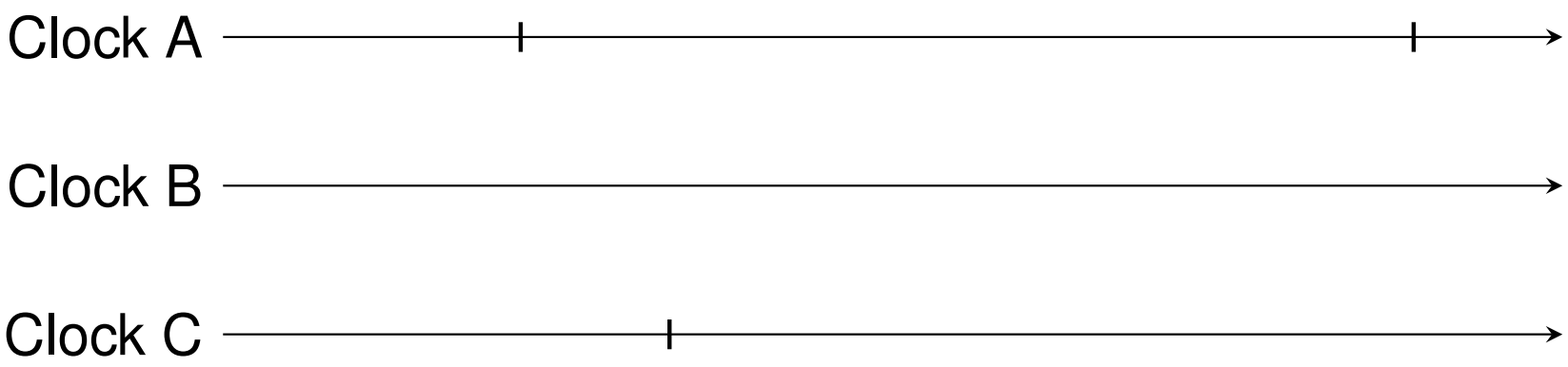
scenario: 1 @2.0; 0 @2.8; 1 @3.0; 0 @5.0; 1 @7.0; 0 @7.4; -1 @10.0; 0 @10.1;

TESL for Modeling Time



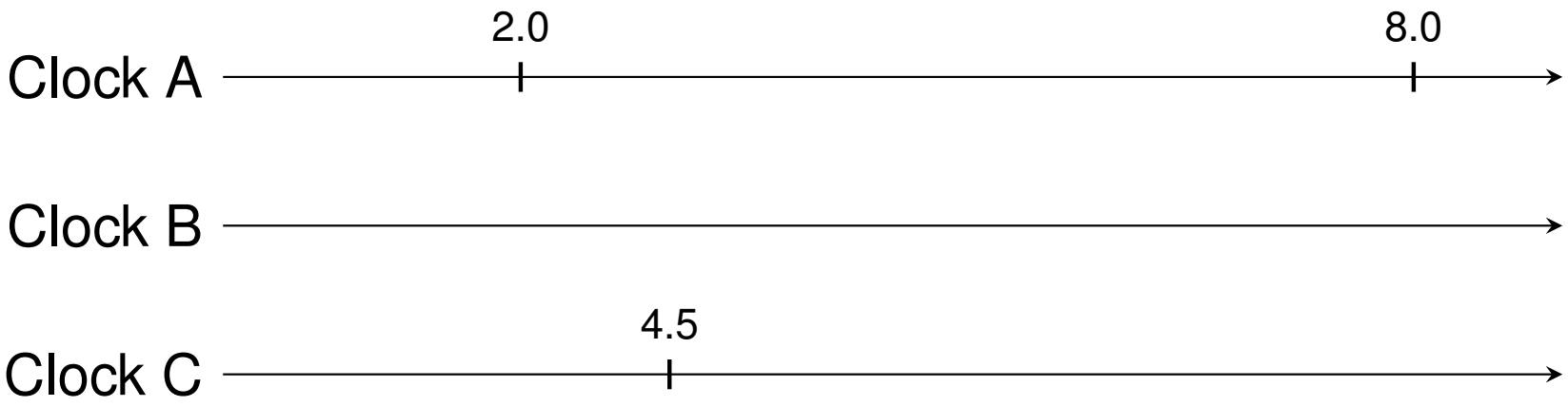
- Clocks model events

TESL for Modeling Time



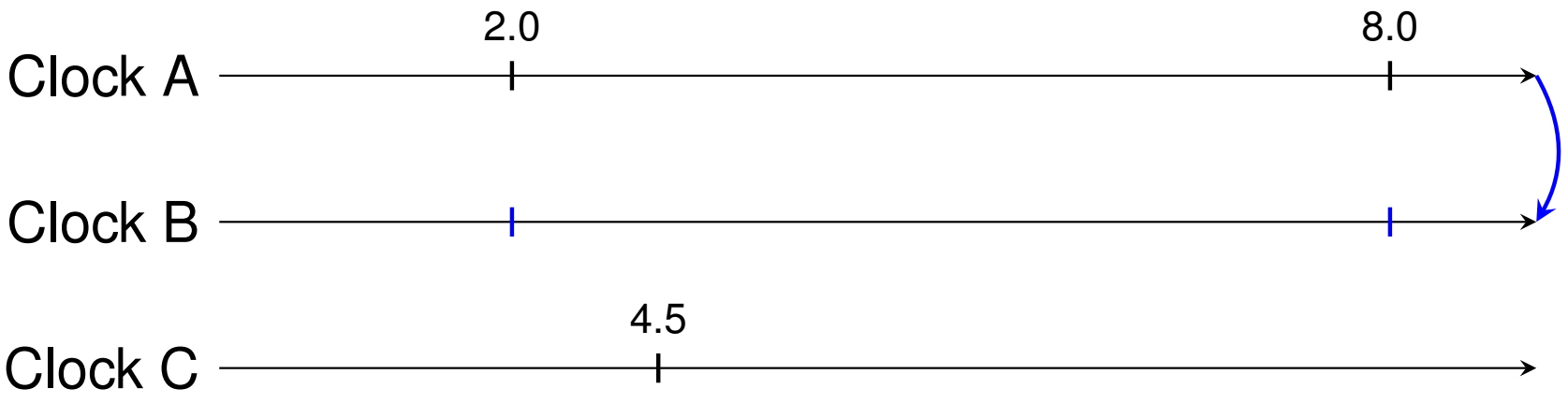
- Clocks model events
- Ticks model event occurrences

TESL for Modeling Time



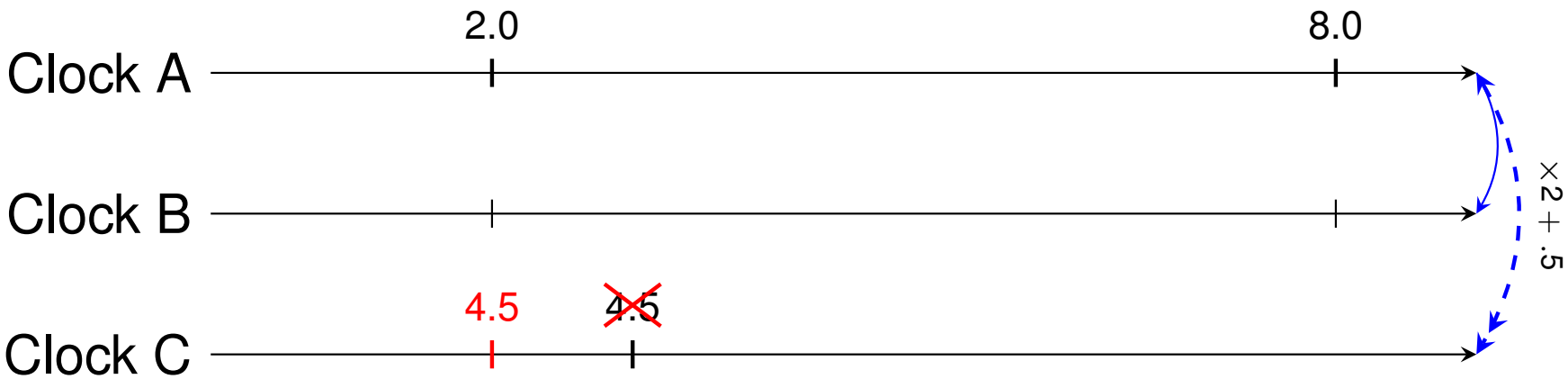
- Clocks model events
- Ticks model event occurrences
- Ticks may have a time tag

TESL for Modeling Time



- Clocks model events
- Ticks model event occurrences
- Ticks may have a time tag
- Implication relations model instantaneous causality

TESL for Modeling Time



- Clocks model events
- Ticks model event occurrences
- Ticks may have a time tag
- Implication relations model instantaneous causality
- Tag relations link time scales and synchronize events

Modeling Time and Control in the Power Window

Clocks are used for modeling the control of different parts of the model

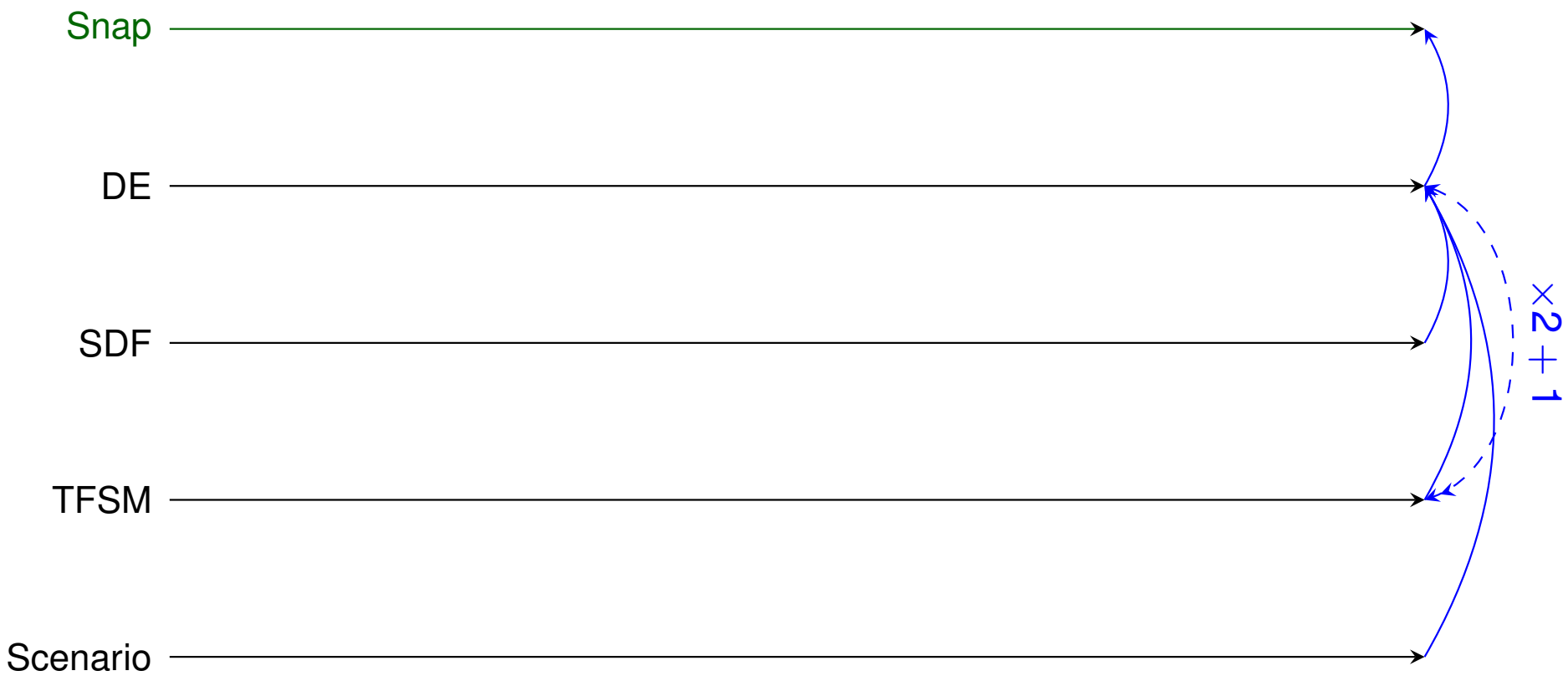
Modeling Time and Control in the Power Window

Control for the top level model implies control in the simulation

Modeling Time and Control in the Power Window

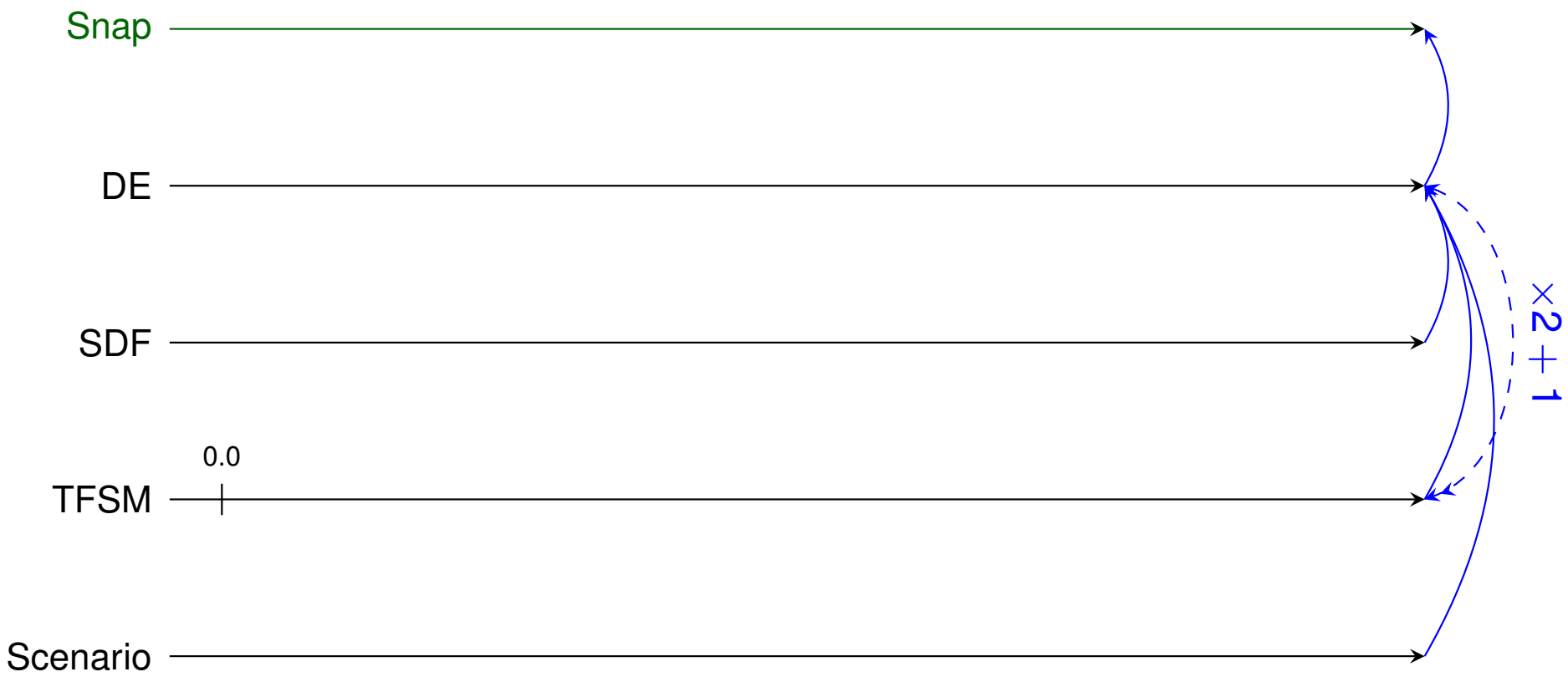
Control for embedded models imply control for the embedding model

Modeling Time and Control in the Power Window



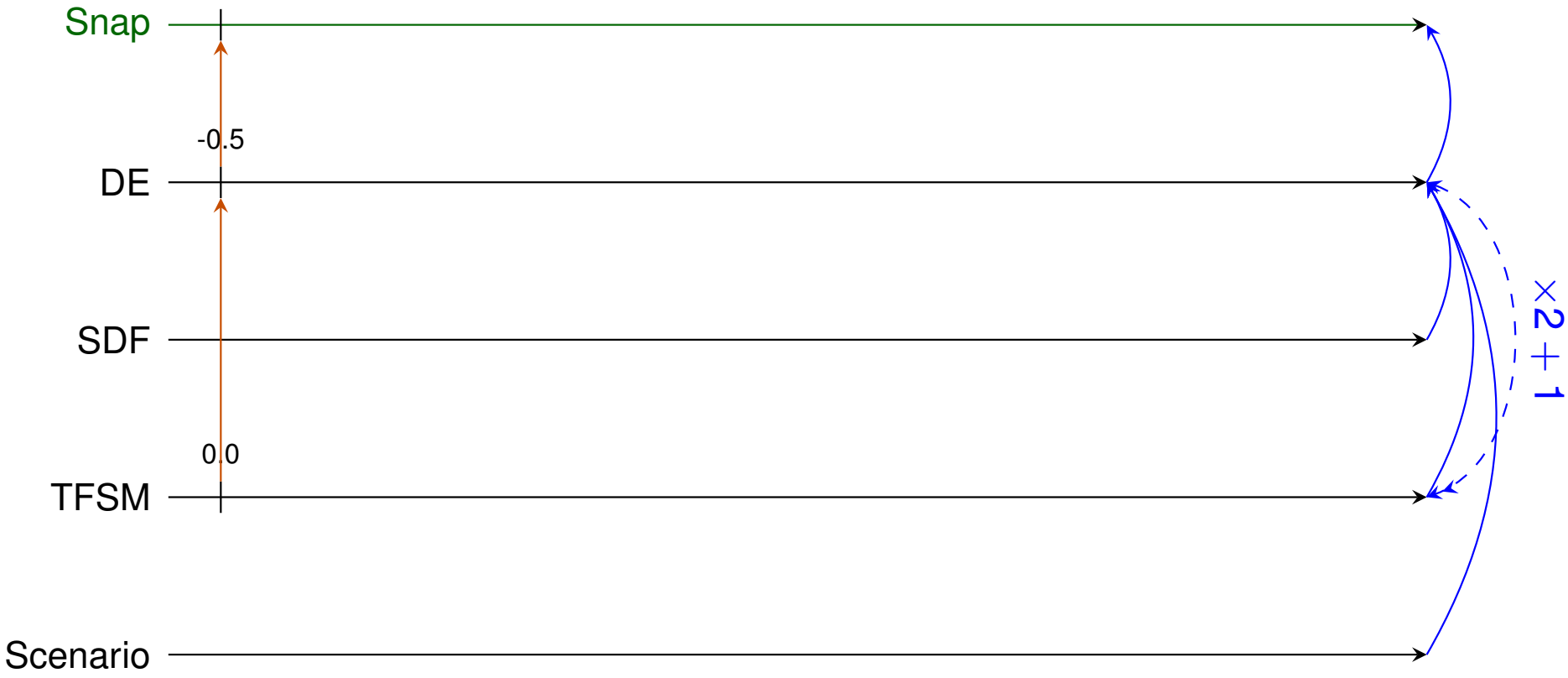
Time in TFSM runs twice as fast as in DE and is offset by one

Modeling Time and Control in the Power Window



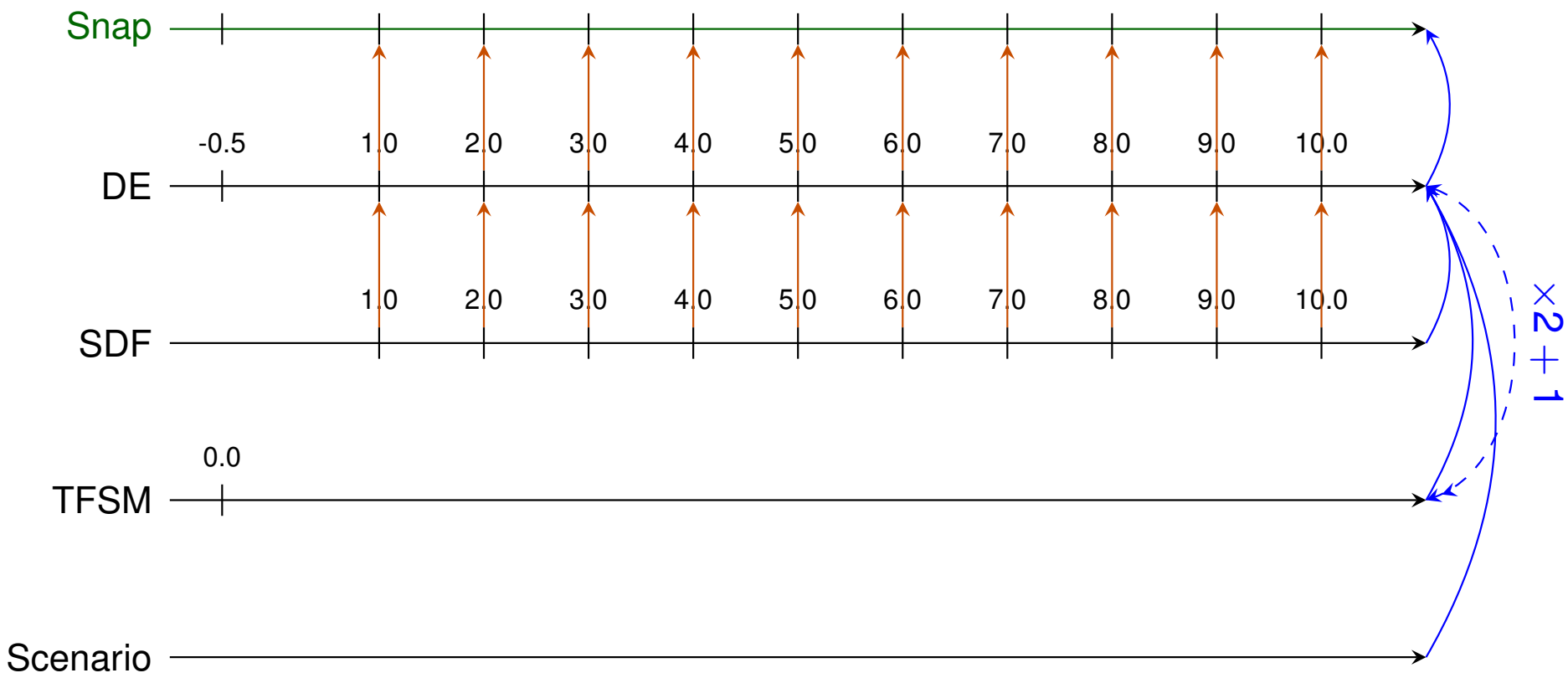
TFSM must receive control at 0.0

Modeling Time and Control in the Power Window



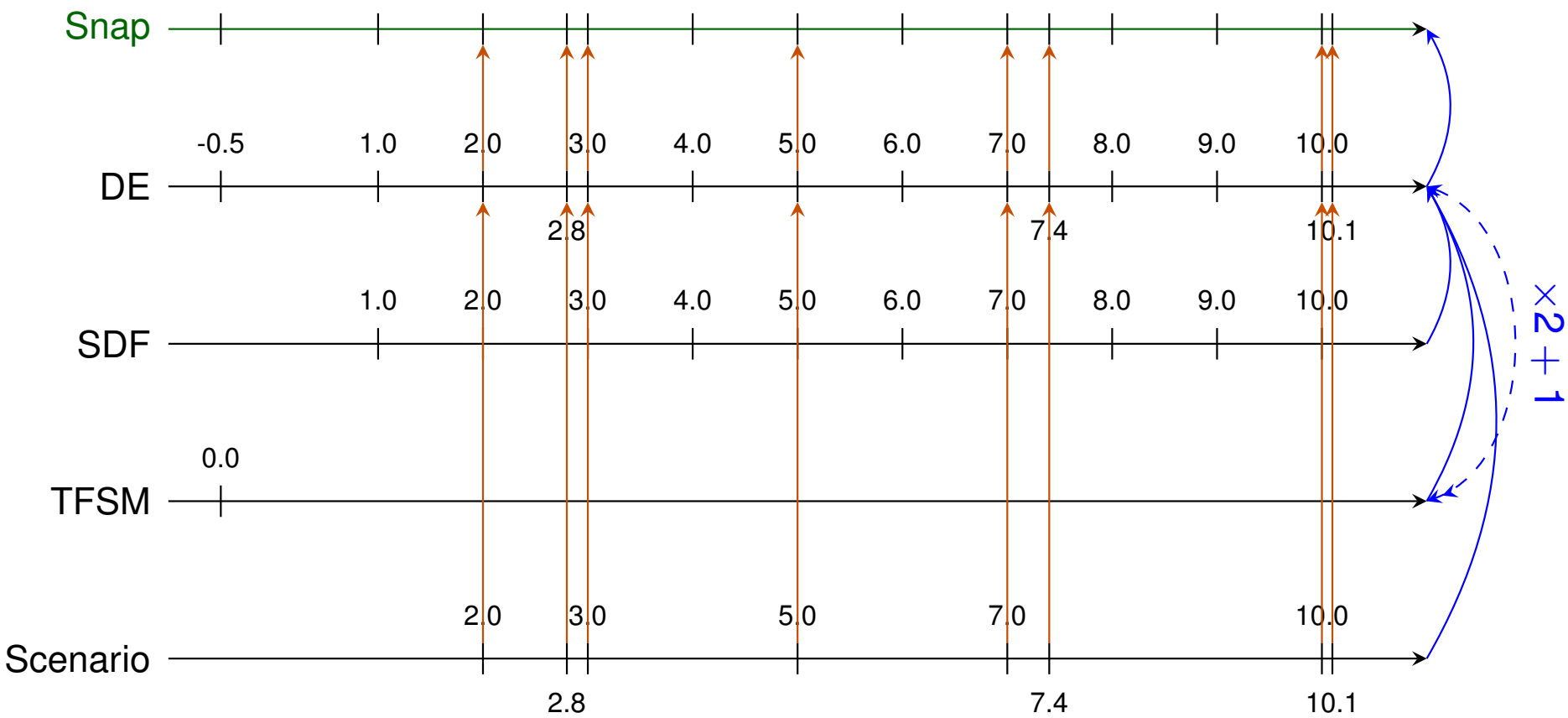
So there must be control in DE at -0.5

Modeling Time and Control in the Power Window



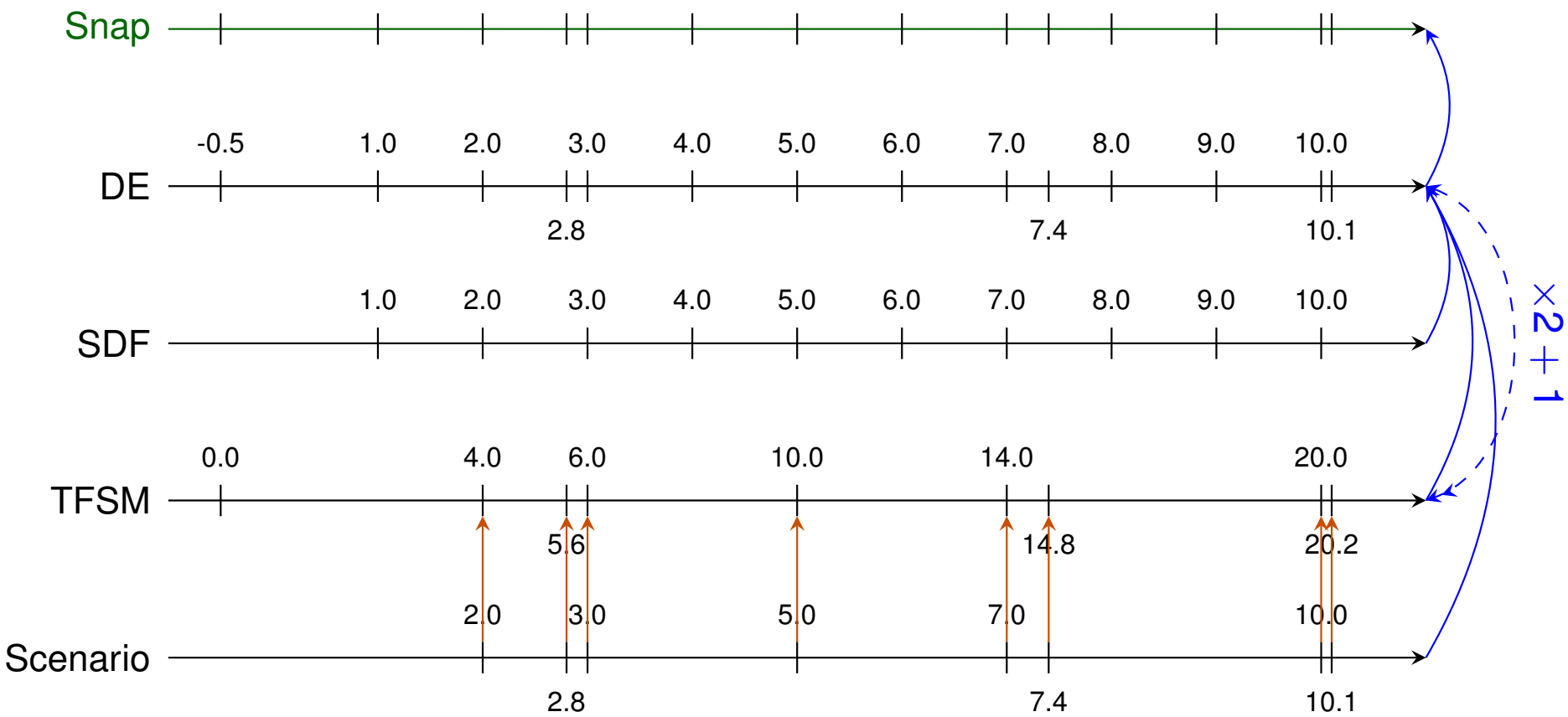
SDF must receive control with period 1.0 on DE time starting at 1.0

Modeling Time and Control in the Power Window



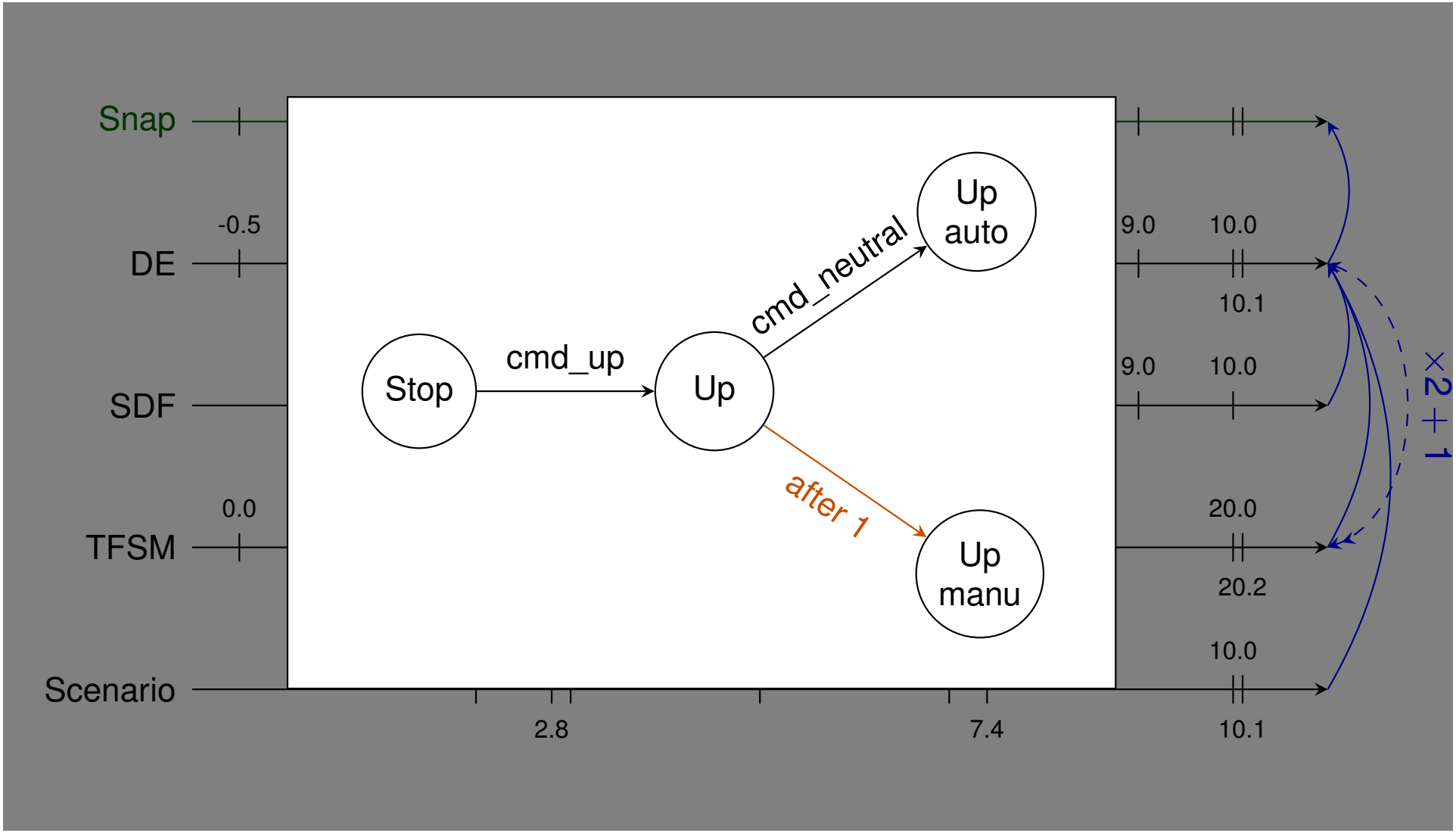
The scenario block must receive control to play its scenario

Modeling Time and Control in the Power Window



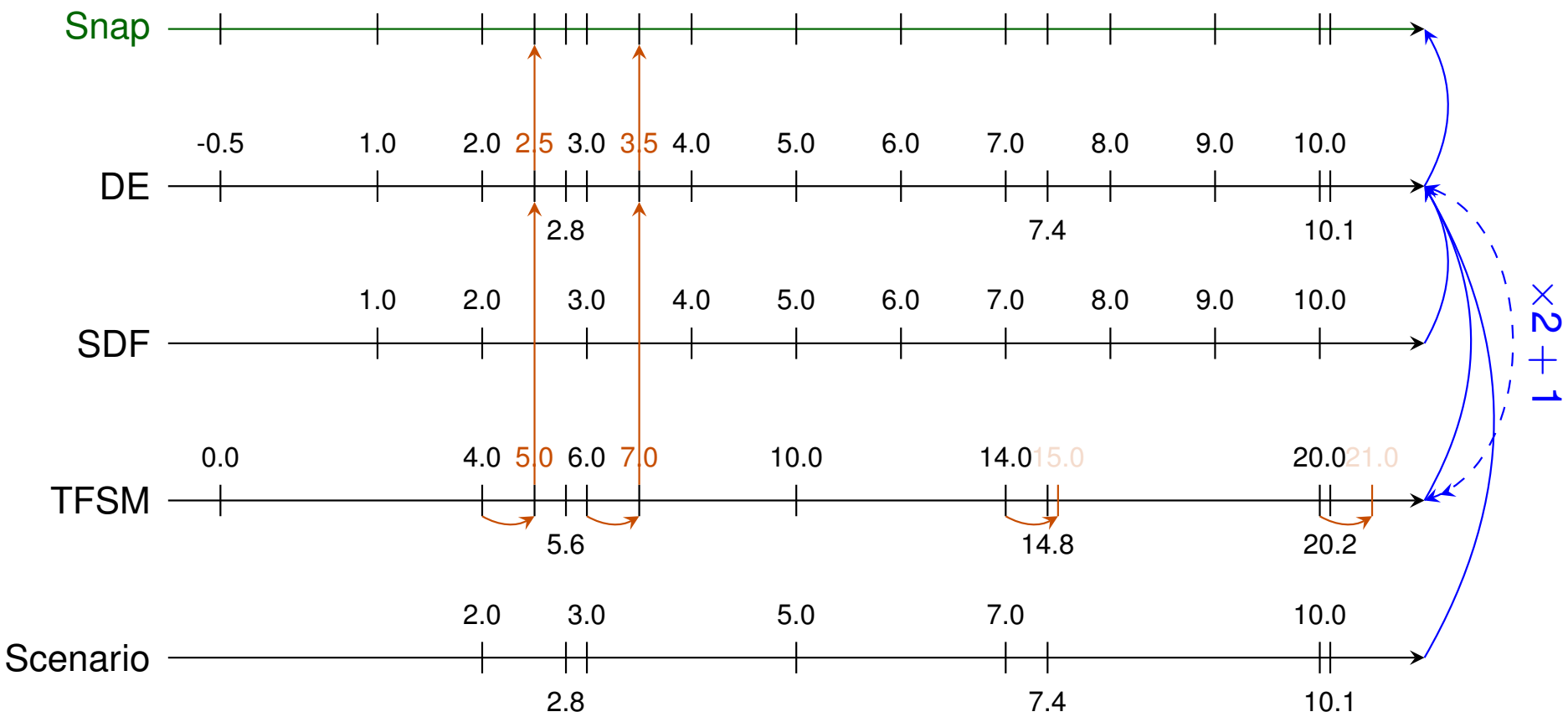
DE semantics creates control on the TFSM IB when it receives inputs

Modeling Time and Control in the Power Window



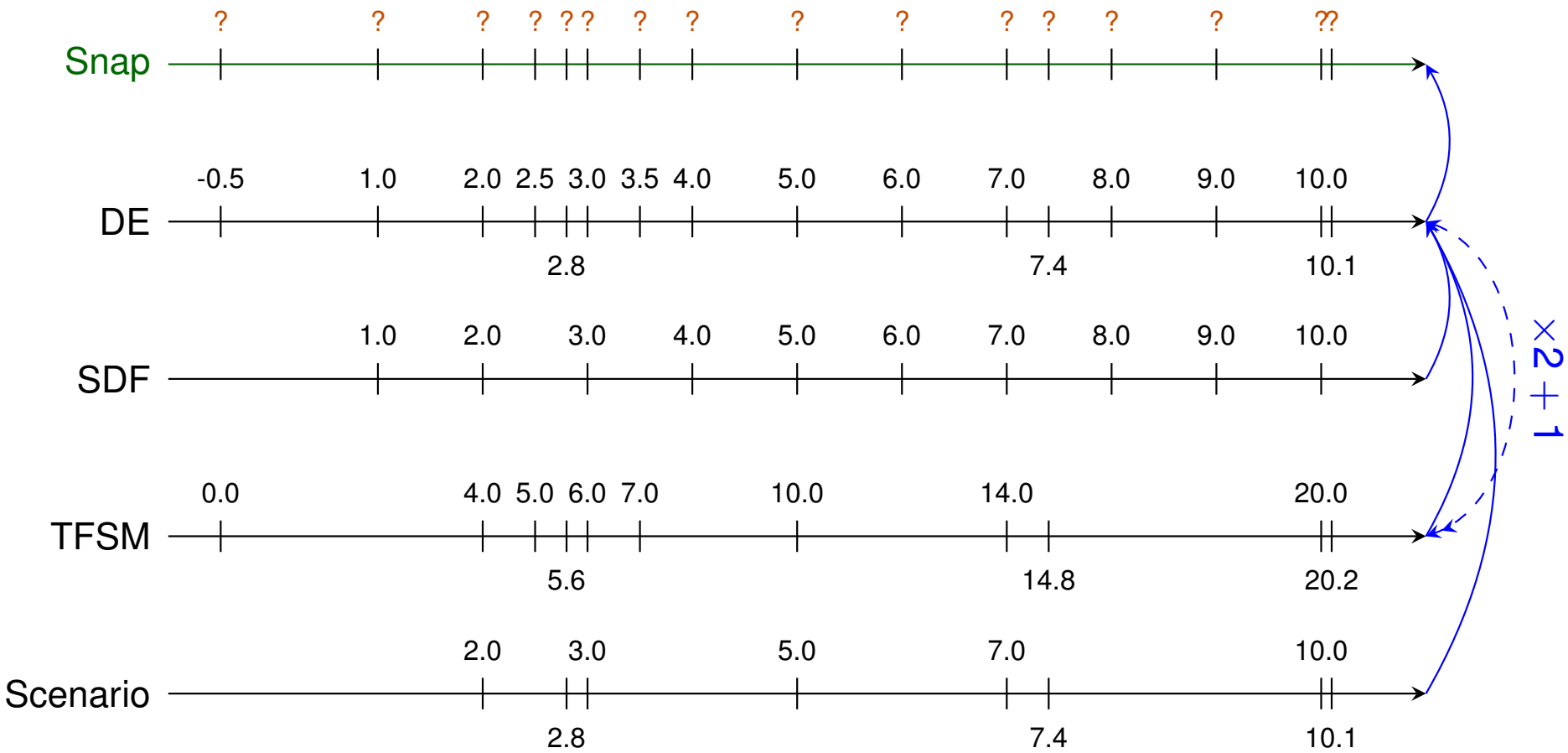
The TFSM controller model has timed transitions

Modeling Time and Control in the Power Window



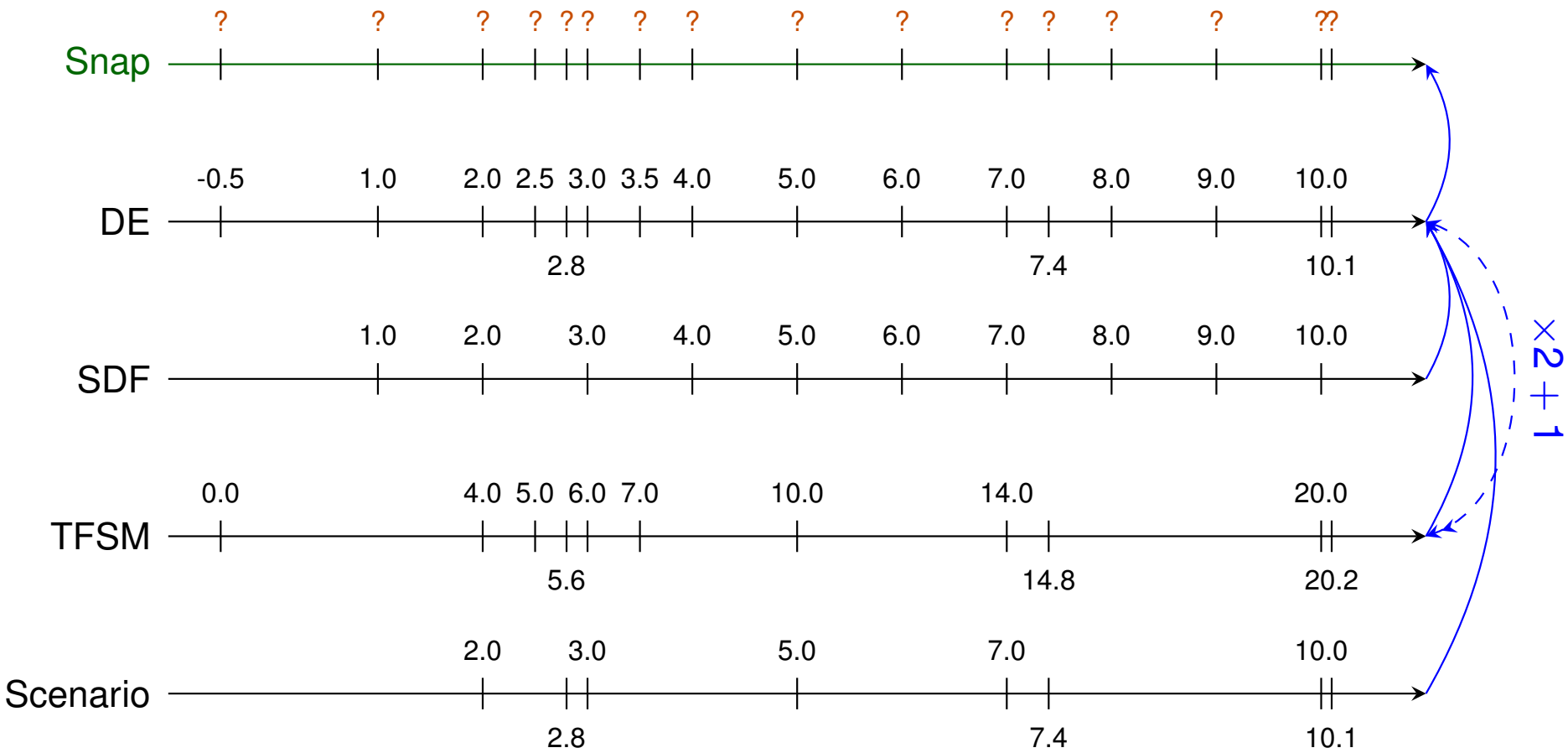
TFSM semantics creates control in TFSM for timed transitions

Modeling Time and Control in the Power Window



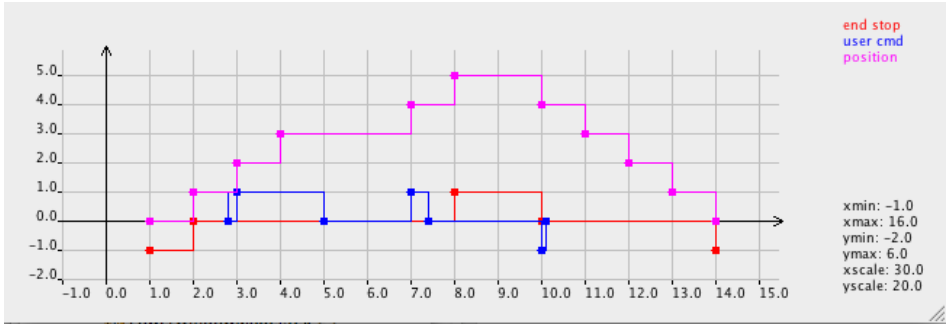
When do these events occur?

Modeling Time and Control in the Power Window



When do these events occur? Driving clocks drive the simulation.

Demo


end stop
user cmd
position
5.0
4.0
3.0
2.0
1.0
0.0
-1.0
-2.0
-1.0 0.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0
xmin: -1.0
xmax: 16.0
ymin: -2.0
ymax: 6.0
xscale: 30.0
yscale: 20.0


Obstacle
Tinted windows

Discussion

Simulation vs. Specification

The engine of ModHel'X is for simulation

- Computes one possible behavior
- Relies on TESL instead of CCSL which has:
  - More powerful time operators (and more complex solver)
  - No support for arbitrary time tags
- Cannot be used for verification (but OK for running tests)

Discussion

Simulation vs. Specification

The engine of ModHel'X is for simulation

- Computes one possible behavior
- Relies on TESL instead of CCSL which has:
  - More powerful time operators (and more complex solver)
  - No support for arbitrary time tags
- Cannot be used for verification (but OK for running tests)

API vs. Modeling Language

ModHel'X models and semantic adaptation are built using an API

- Tight coupling between semantics and implementation
- Ongoing work for modeling semantic adaptation (MPM paper)
- Ongoing work in Gemoc for a higher level description of MoCs

Conclusion

Contribution

- A complete model with explicit semantic adaptation
- A model of time for semantic adaptation of time and control
- Available on wwwdi.supelec.fr/software/ModHelX/PowerWindow

Conclusion

Contribution

- A complete model with explicit semantic adaptation
- A model of time for semantic adaptation of time and control
- Available on wwwdi.supelec.fr/software/ModHeIX/PowerWindow

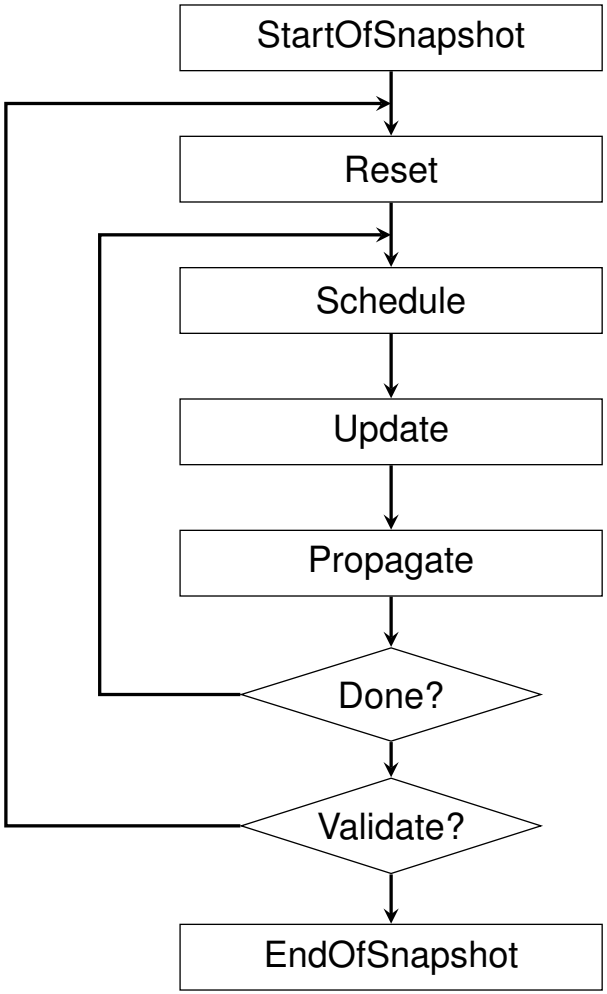
Perspectives

- Ongoing work for a declarative definition of semantic adaptation
- Future work: defining MoCs using hierarchical TESL specs
- Future work: conformance of a TESL spec to a CCSL spec

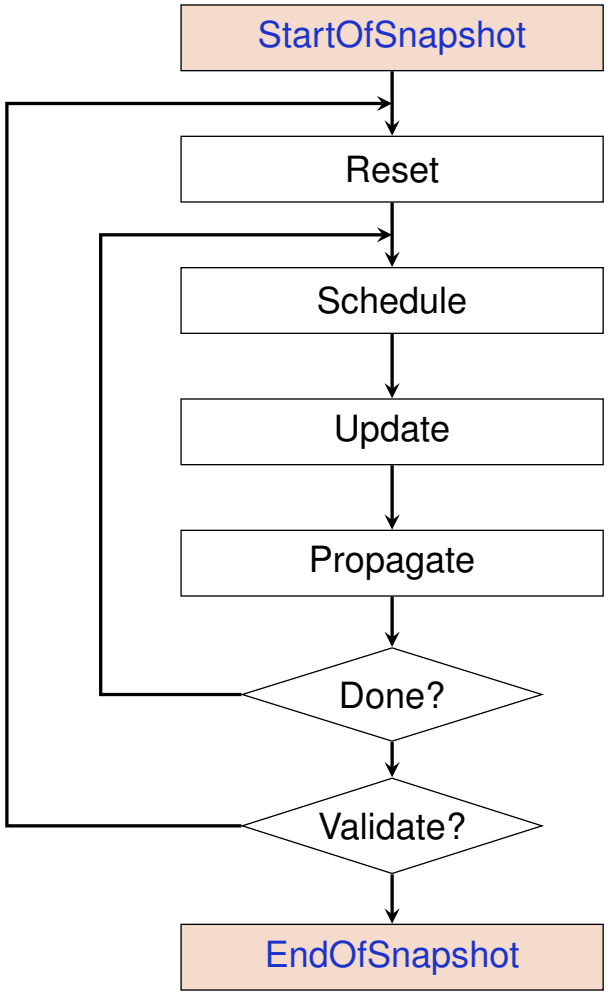
Questions?

Thanks for your attention

ModHel'X: Generic Execution Engine



ModHel'X: Generic Execution Engine

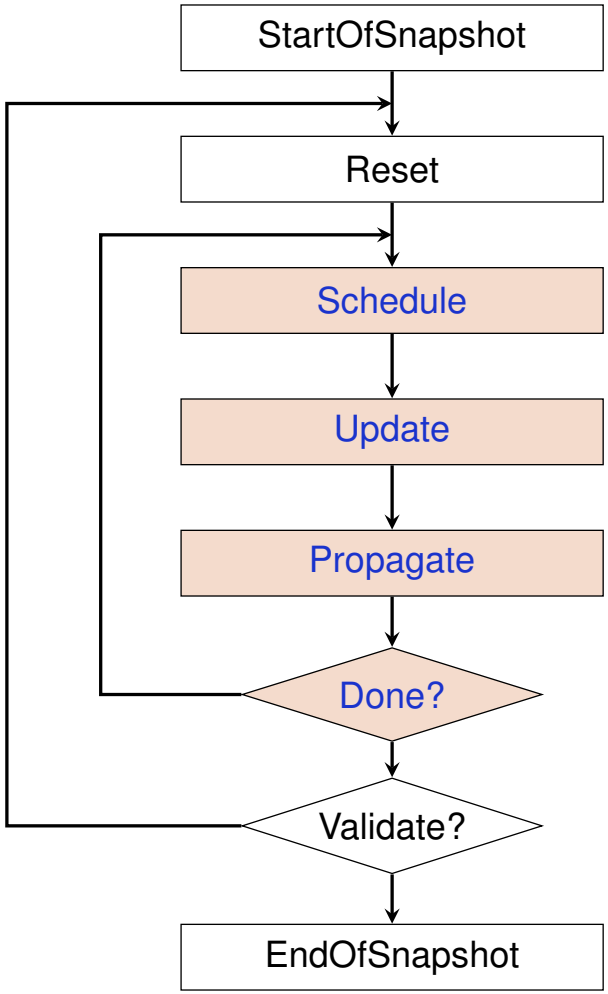


Behavior of a model = series of observations

Synchronous approach of the observation of models:

- no communication with the environment during the snapshot;
- no modification of the internal state of the blocks.

ModHel'X: Generic Execution Engine



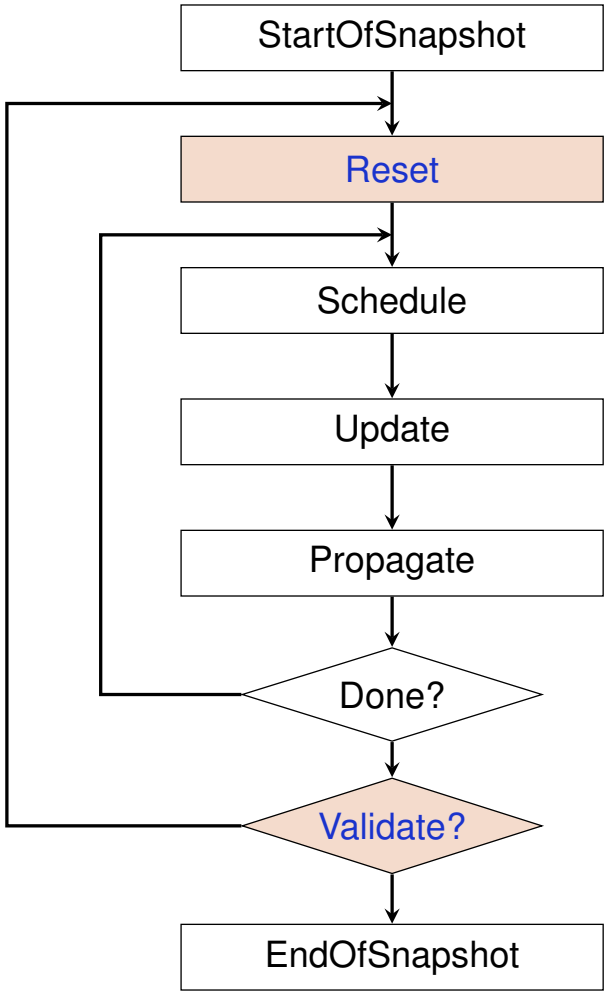
Computation of a fixed point by iteration:

- sequential observation of the blocks;
- update of their interface;
- propagation of the information according to the relations between pins.

Schedule, Propagate and Done are the 3 operations which define a MoC.

Update represents the observable behavior of the blocks.

ModHel'X: Generic Execution Engine



It is possible to reject the fixed point that has been reached.

Search another fixed point starting from different initial conditions.