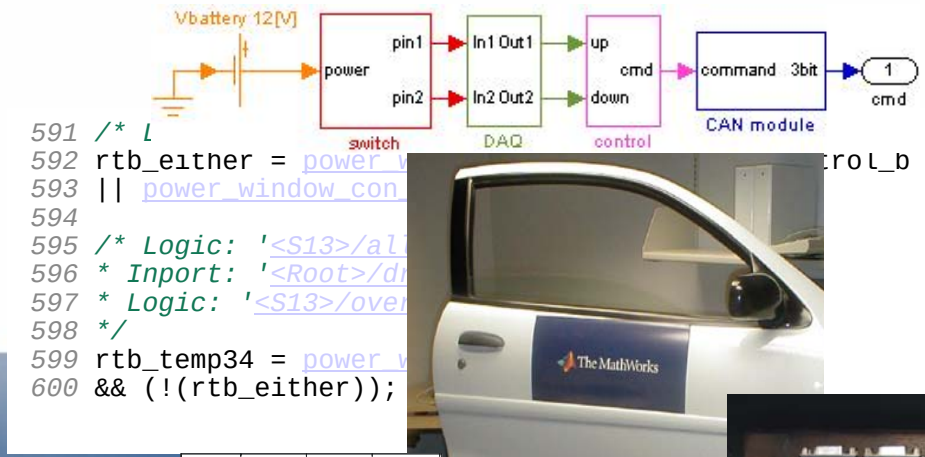
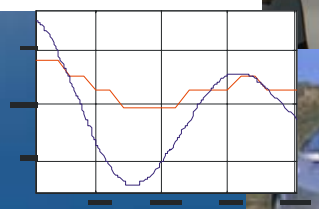
Model-Based Design – What it is and what it still needs





Pieter J. Mosterman
pieter.mosterman@mathworks.com

Senior Research Scientist
The MathWorks

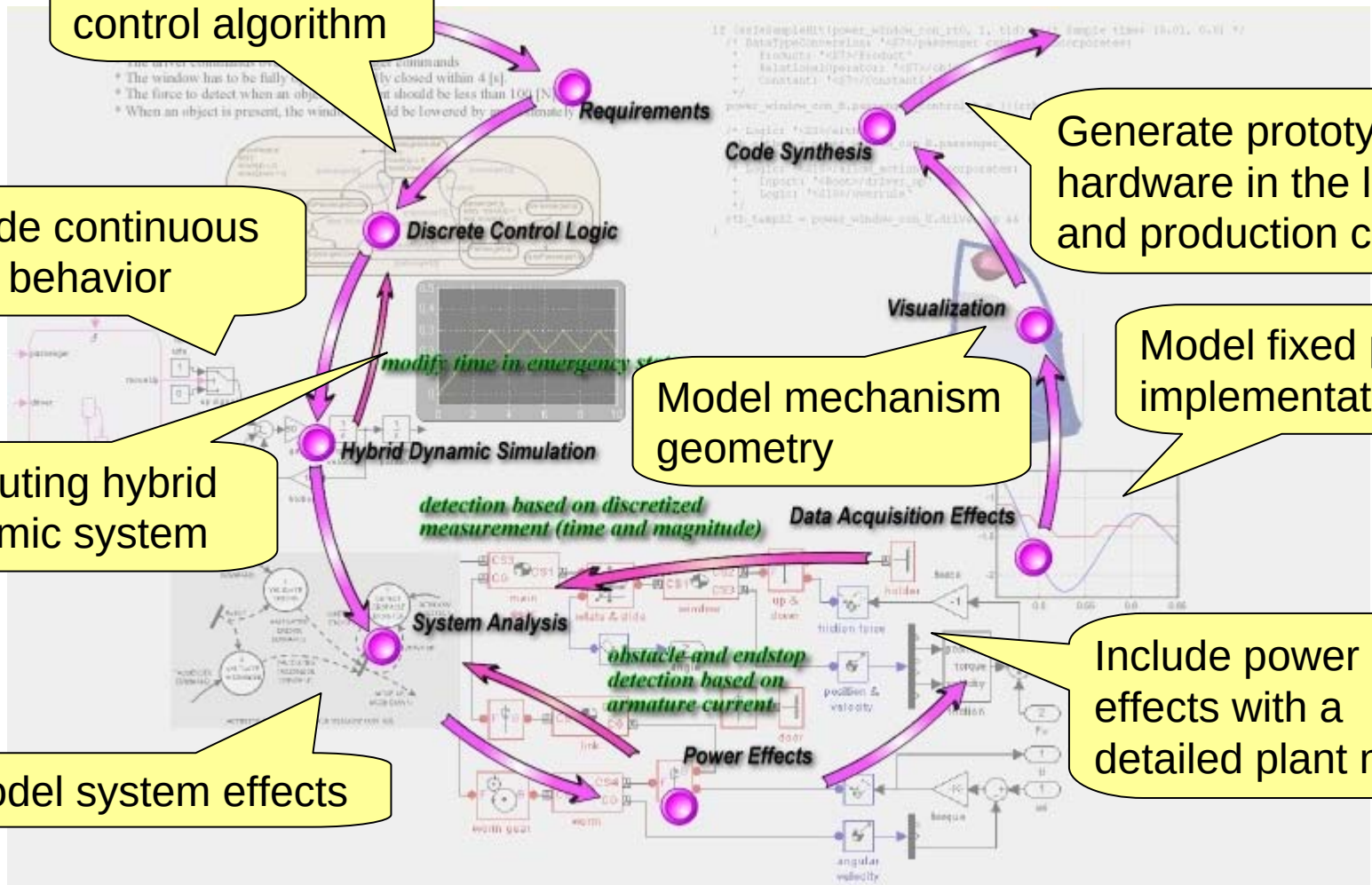
Introduction

- Model-Based Design
  - Exploit computational models
  - Increasingly adopted in industry
- Underlying needs
  - Many different modeling formalisms
    - Syntax
    - Semantics
  - Relate and combine models
    - Different parts of a system
    - Different design stages of a system
- Open research topics …

Agenda

- Model-Based Design
- Computer Automated Multiparadigm Modeling
- Application
- Multi-formalism modeling
- Mixed-signal simulation
- Hybrid Dynamic Systems
- Summary

Power Window Demo Summary



Model discrete control algorithm

Include continuous plant behavior

Executing hybrid dynamic system

Model system effects

Model mechanism geometry

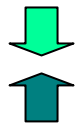
Model fixed point implementation

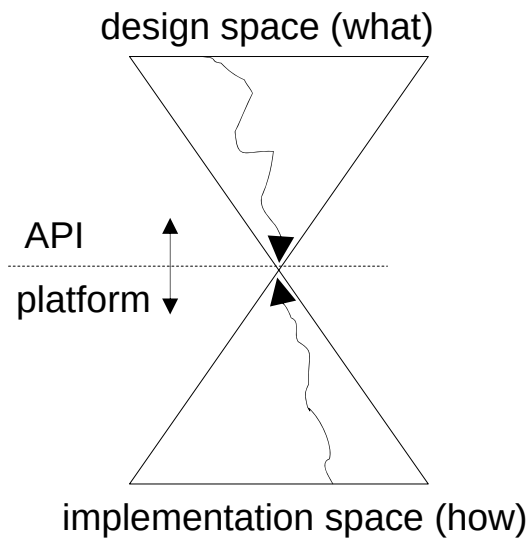
Generate prototype, hardware in the loop, and production code

Include power effects with a detailed plant model

Design?

- Design of complex systems is a process of transaction
  - Requirements (what)
  - Specification (how)
- Platform-based design
  - Meet in the middle



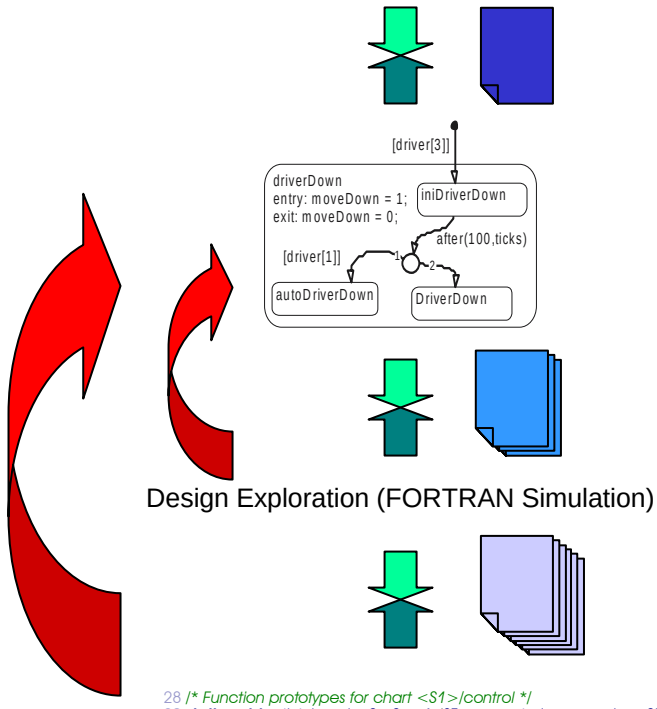


Typical Document-Based Design

- Results from each design stage are documented
- Specifications output of one stage are requirements input to the next
- Simulation requires programming of specification (e.g., FORTRAN)

The window has to be fully opened and fully closed within 4 [s].

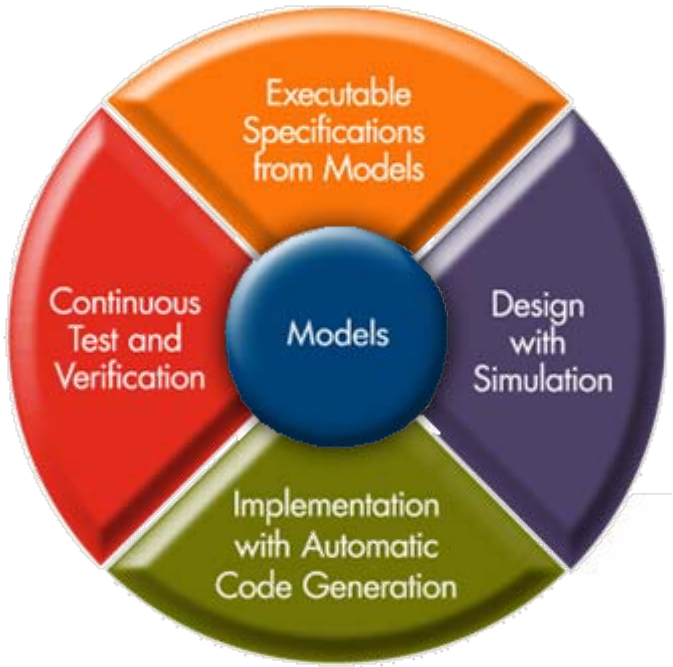
The force to detect when an object is present should be less than 100 [N].




```
28 /* Function prototypes for chart <S1>/control */
29 static void exit_internal_c2_s2_safe(SFpower_window_con_rtw_c2InstanceStruct
30 *chartInstance);
34
35 #define IN_NO_ACTIVE_CHILD (0)
36 #define IN_c2_s1_emergencyDown 1
39 #define IN_c2_s7_driverNeutral 2
591 /* Logic: '<S3>/either' */
592 rtb_either = power_window_con_B.passenger_control_b
593 || power_window_con_B.passenger_control_a;
```

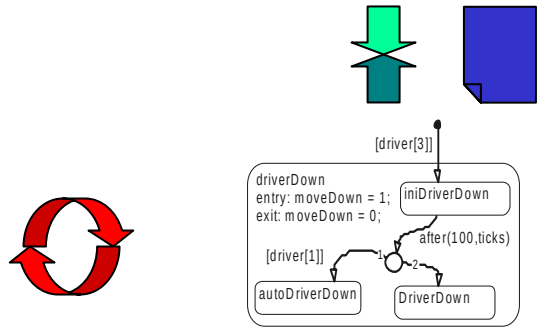
Model-Based Design

- Share in computational form



The window has to be fully opened and fully closed within 4 [s].

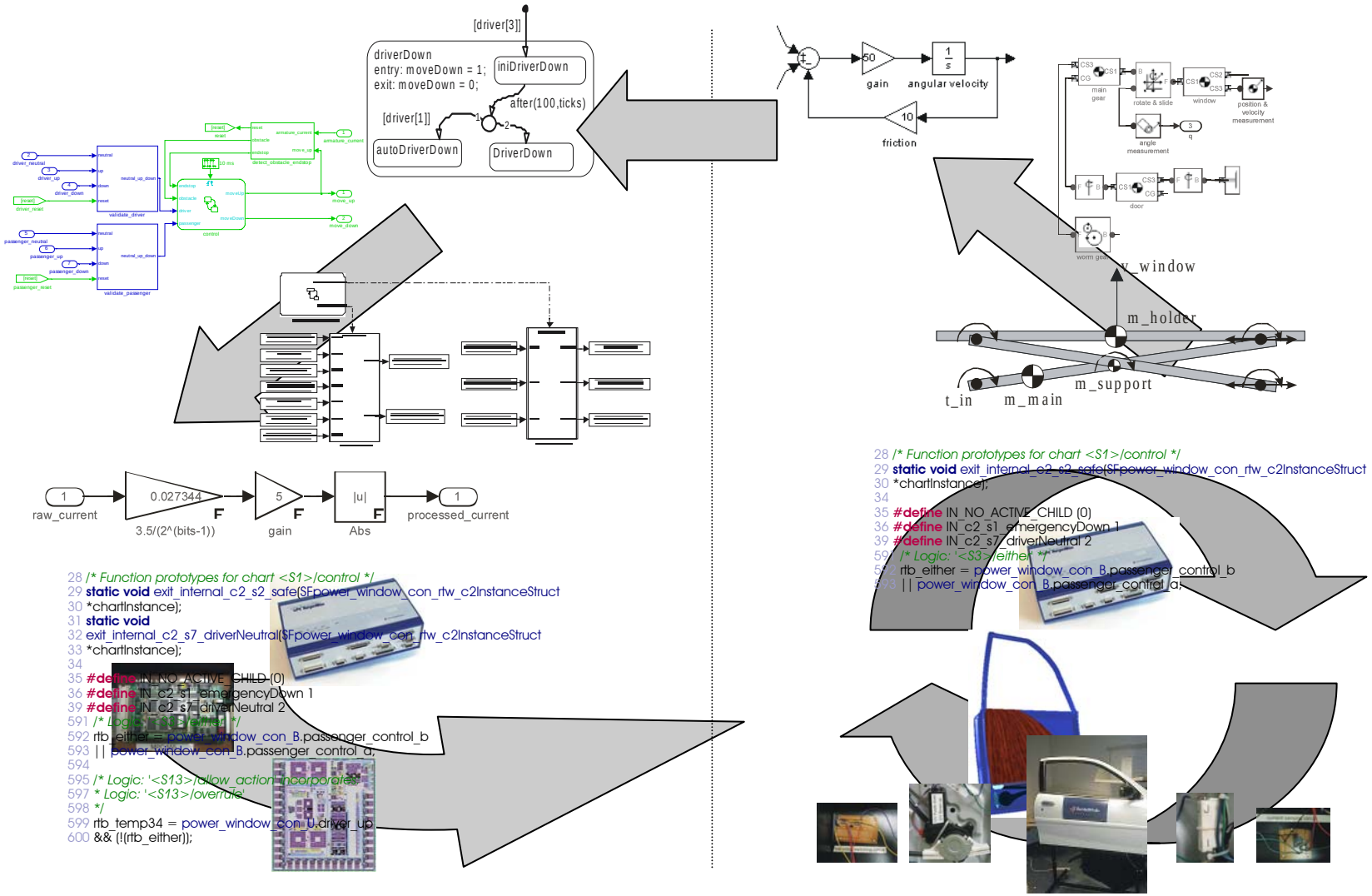
The force to detect when an object is present should be less than 100 [N].



Design Exploration

```
28 /* Function prototypes for chart <S1>/control */
29 static void exit_internal_c2_s2_safe(SFpower_window_con_rtw_c2InstanceStruct
30 *chartInstance);
34
35 #define IN_NO_ACTIVE_CHILD (0)
36 #define IN_c2_s1_emergencyDown 1
39 #define IN_c2_s7_driverNeutral 2
591 /* Logic: '<S3>/either' */
592 rtb_either = power_window_con_B.passenger_control_b
593 || power_window_con_B.passenger_control_a;
```

Where are the models?



Benefits of a computational form

- Simulate a design immediately
  - Try out different ideas quickly
  - Combine with partially detailed design
  - Allow sharing of design decisions across stages
  - Domain-specific languages can easily be developed
- Automatic model processing
  - Style guideline enforcing (automatic fixing)
  - Data type inference
- Static checking of dynamics
- Test bench generation
- Digital linking to other documents
- Model transformation
  - Allows one reference model
  - Choose best representation for given analysis
  - Automate synthesis
  - Automate optimization
  - Automate design?!

Combine semantics!

Style specification; static semantics!

Coverage test suites; prover complexity!

Consistency and inconsistency!

Correctness!

Performance evaluation!

Architecture modeling!

Agenda

- Model-Based Design
- Computer Automated Multiparadigm Modeling
- Application
- Multi-formalism modeling
- Mixed-signal simulation
- Hybrid Dynamic Systems
- Summary

Computer Automated Multiparadigm Modeling

- Computer Automated Multiparadigm Modeling (CAMPaM)
  - Annual McGill Bellairs workshop
- Three elements of CAMPaM
  - Multi-formalism modeling
  - Metamodeling
  - Multiple levels of abstraction
- Model model transformation
  - Graph grammars
  - ...

What is a model anyway?

Jean Bézivin

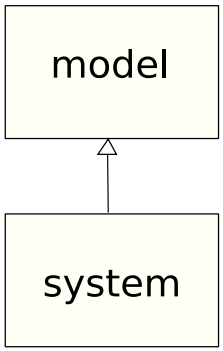
Everything is a model !

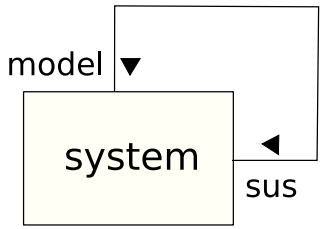
Jean-Marie Favre

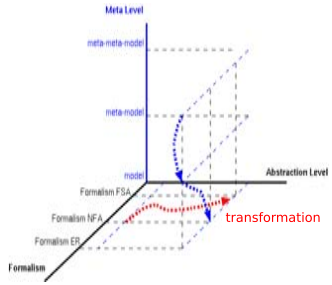
Nothing is a model !

Hans Vangheluwe

Model everything !





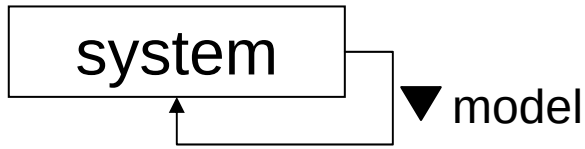


CAMPaM

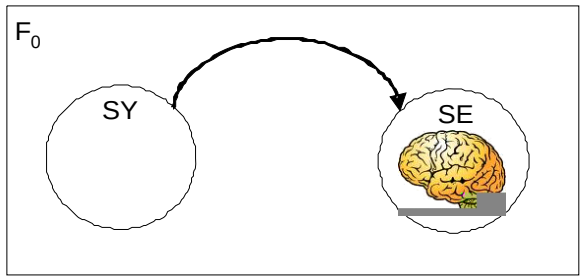
Acknowledgment: *picture, courtesy of Hans Vangheluwe*

Nothing is a model

- A system that represents a system



- Representation in a formalism
- Formally define a formalism (Ogden-Richards triangle of semiotics)
  - Abstract syntax
  - Semantic domain



Syntax versus Semantics

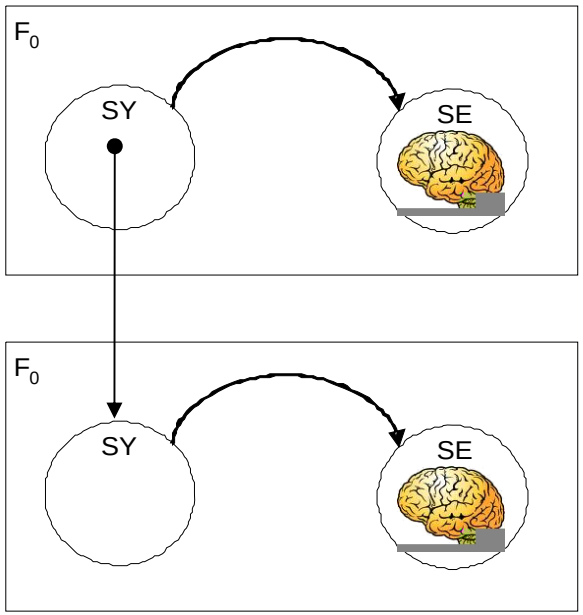
- Fasten seat belt syntax

- Fasten semantics …

- … or open semantics?

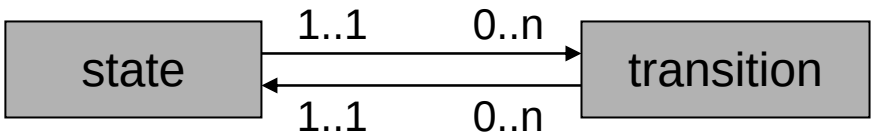
Defining a formalism

- The syntax can be defined by
  - Enumeration
  - A model of the modeling formalism: a ‘meta’ model
    - For example, a grammar

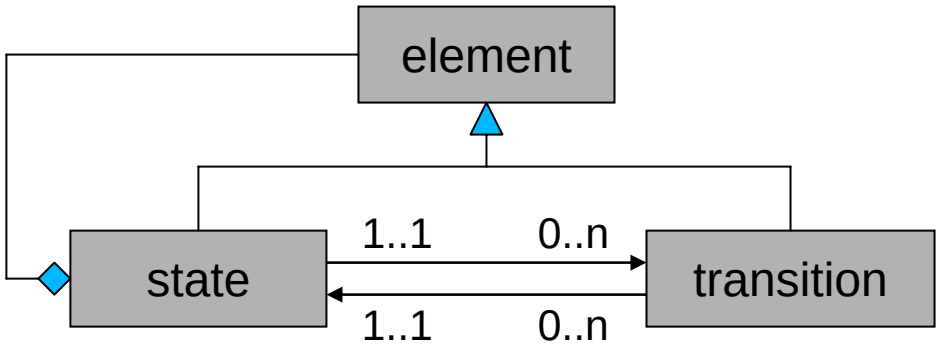


A metamodel

- State transition diagram

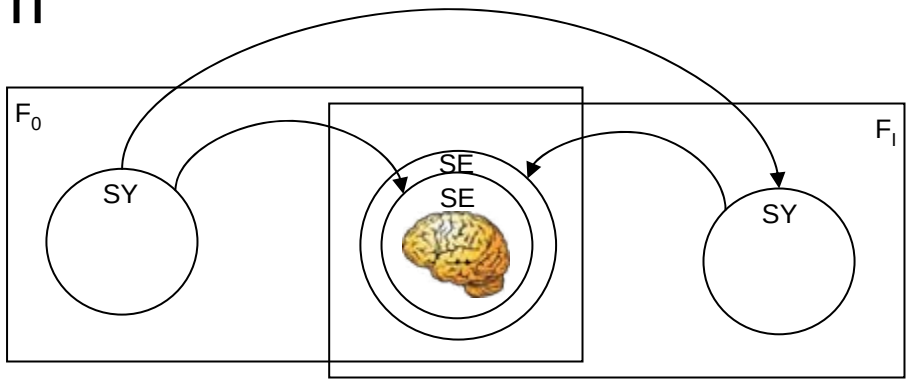


- Automatically generate an editor
- Add hierarchy and regenerate new editor



Defining semantics

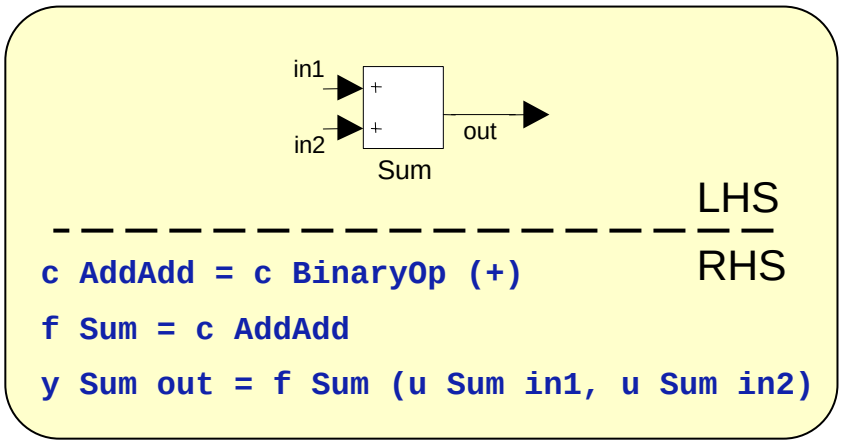
- Define semantics of a new formalism by mapping the abstract syntax onto defined formalism
- The semantic domain to be defined needs to be subsumed by the semantic domain of the defining formalism

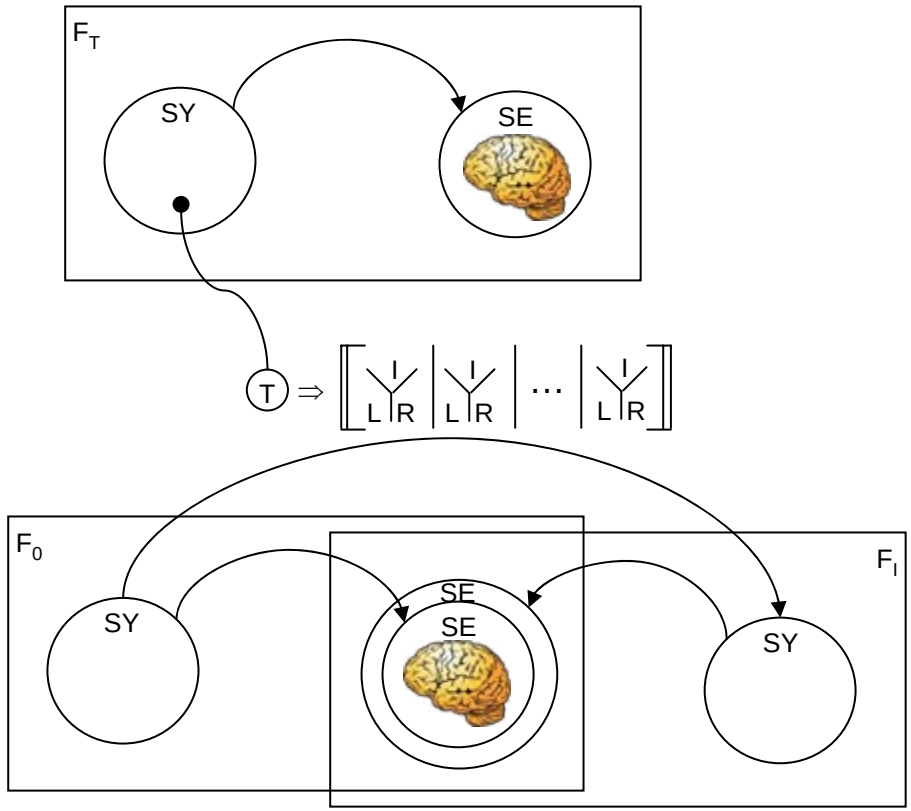


- Model the transformations!
  - Graph rewriting

Model everything

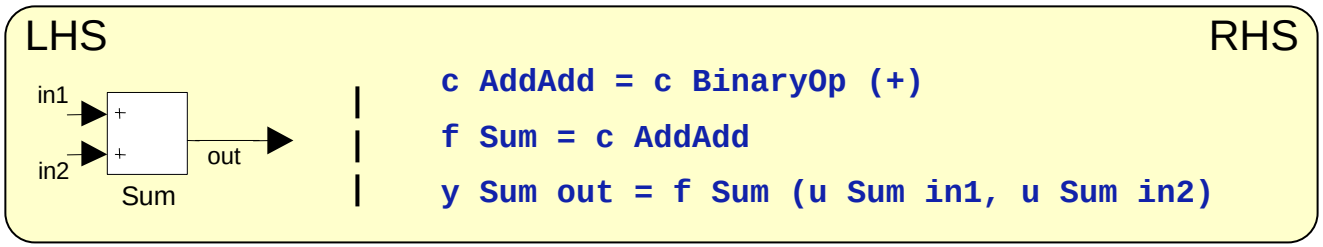
- Reasoning vs. efficiency
  - Operationally
  - Denotationally
- Semantic anchoring
  - Abstract state machines
  - DEVS
  - Haskell



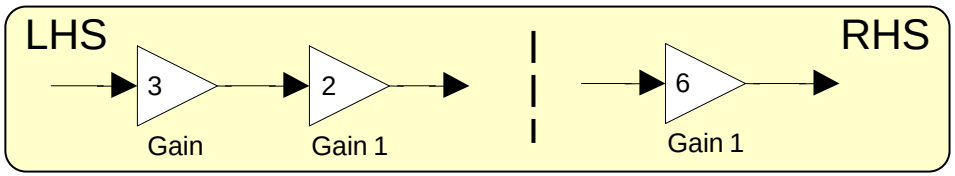


Transformations in general

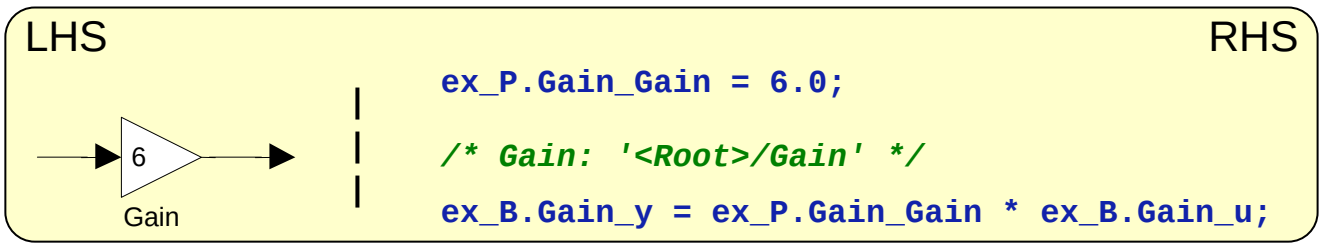
- Defining semantics (Haskell)



- Optimization (minimize number of blocks)



- Code generation



Research

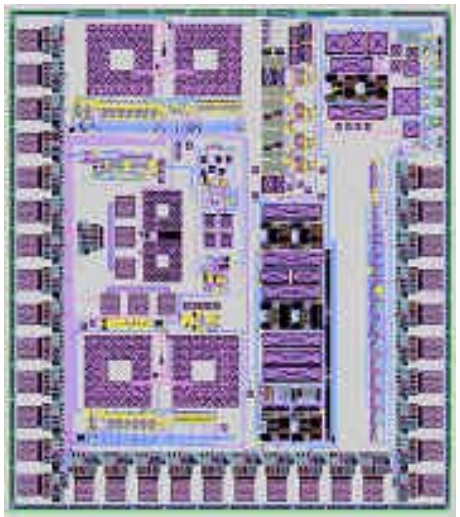
- Provably correct transformations
- Efficiency
  - Programmed graph rewriting
- Domain-specific formalisms evolve constantly
- Semantic anchoring
  - What formalism to use?
  - Is there a metaformalism?
  - How do you deal with semantic nuances?
    - Approximative semantics?

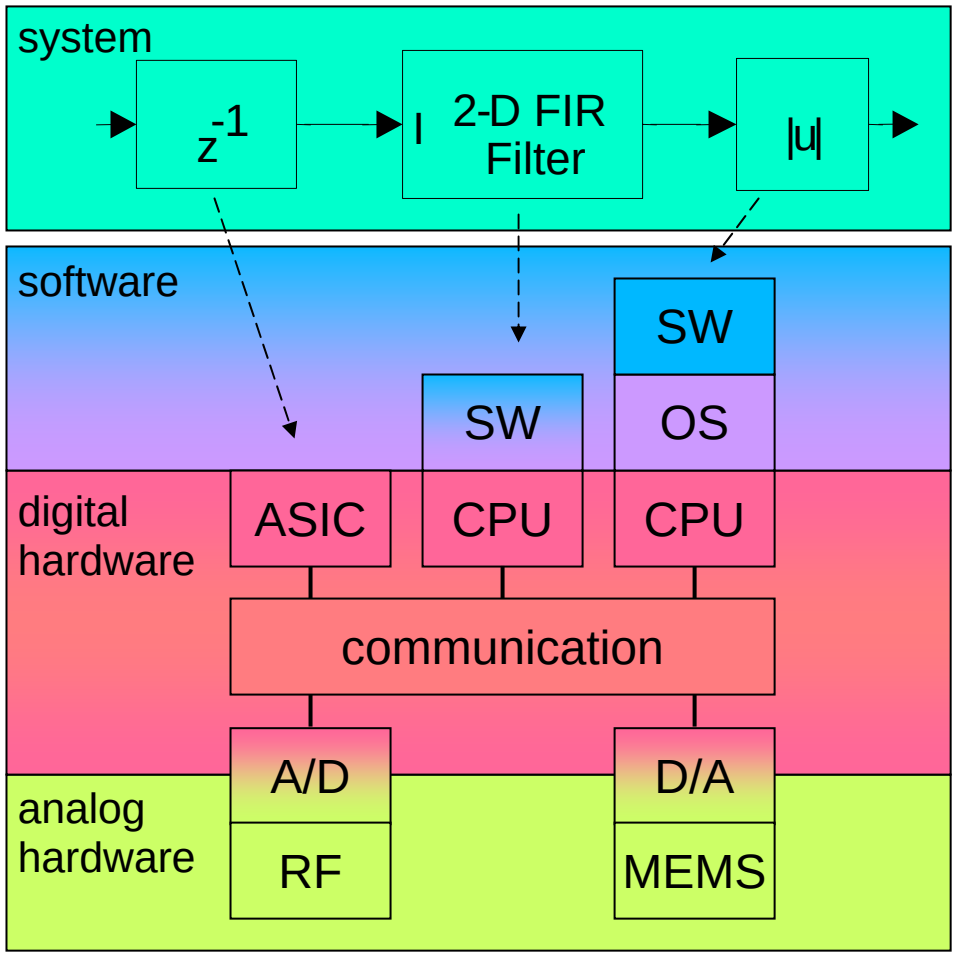
Agenda

- Model-Based Design
- Computer Automated Multiparadigm Modeling
- Application
- Multi-formalism modeling
- Mixed-signal simulation
- Hybrid Dynamic Systems
- Summary

An EDA application

- SoC platform
  - Heterogeneous
  - Highlights more common design challenges

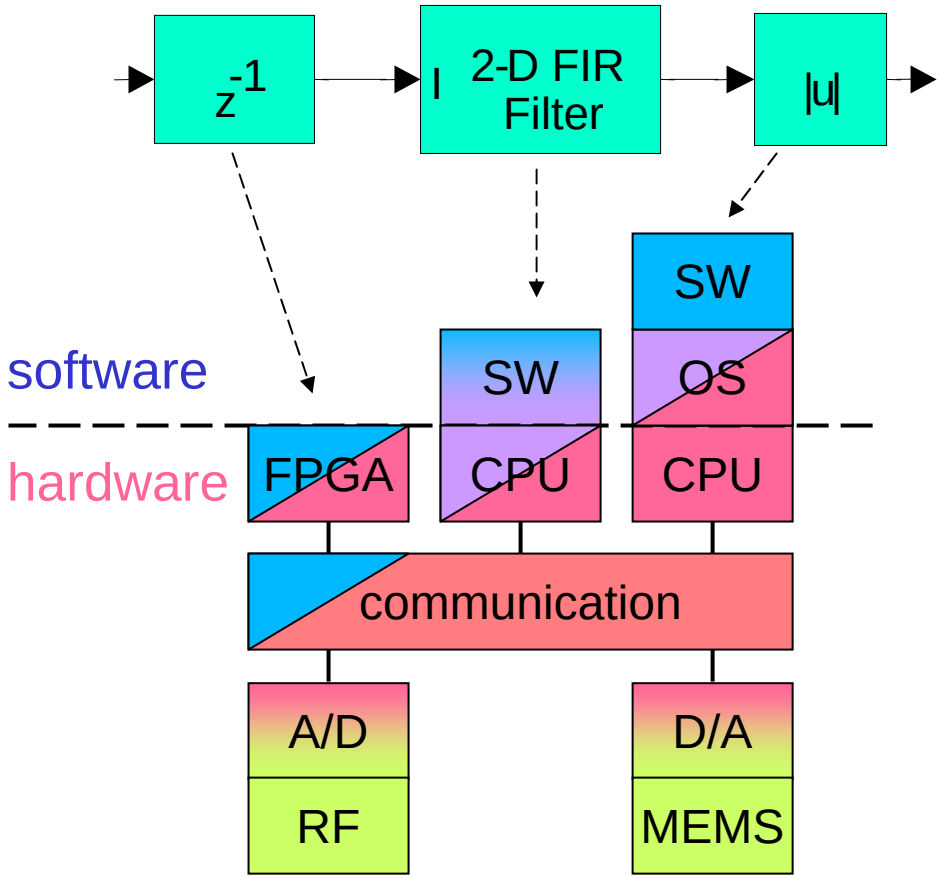




Acknowledgment: *picture redrawn from work by Jan Madsen*

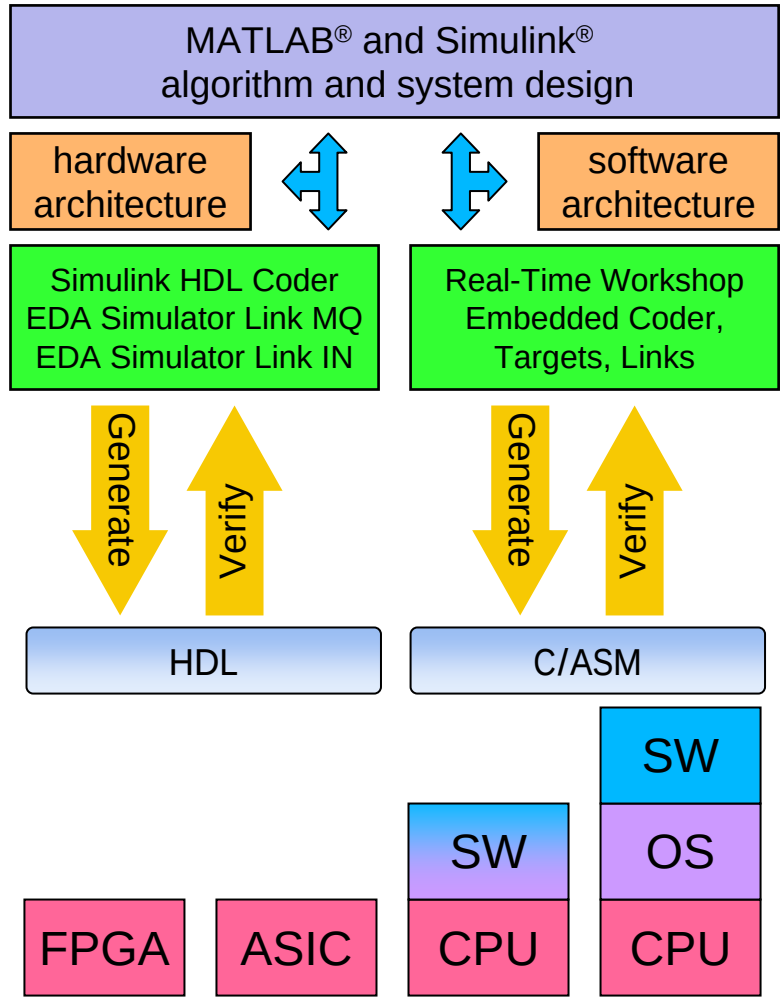
A new frontier …

- No clear HW/SW separation
- Traditionally different design paradigms
- Reconfigurable
  - Hardware and software
  - Adapt to environment
- Novel
  - Design paradigms
  - Applications



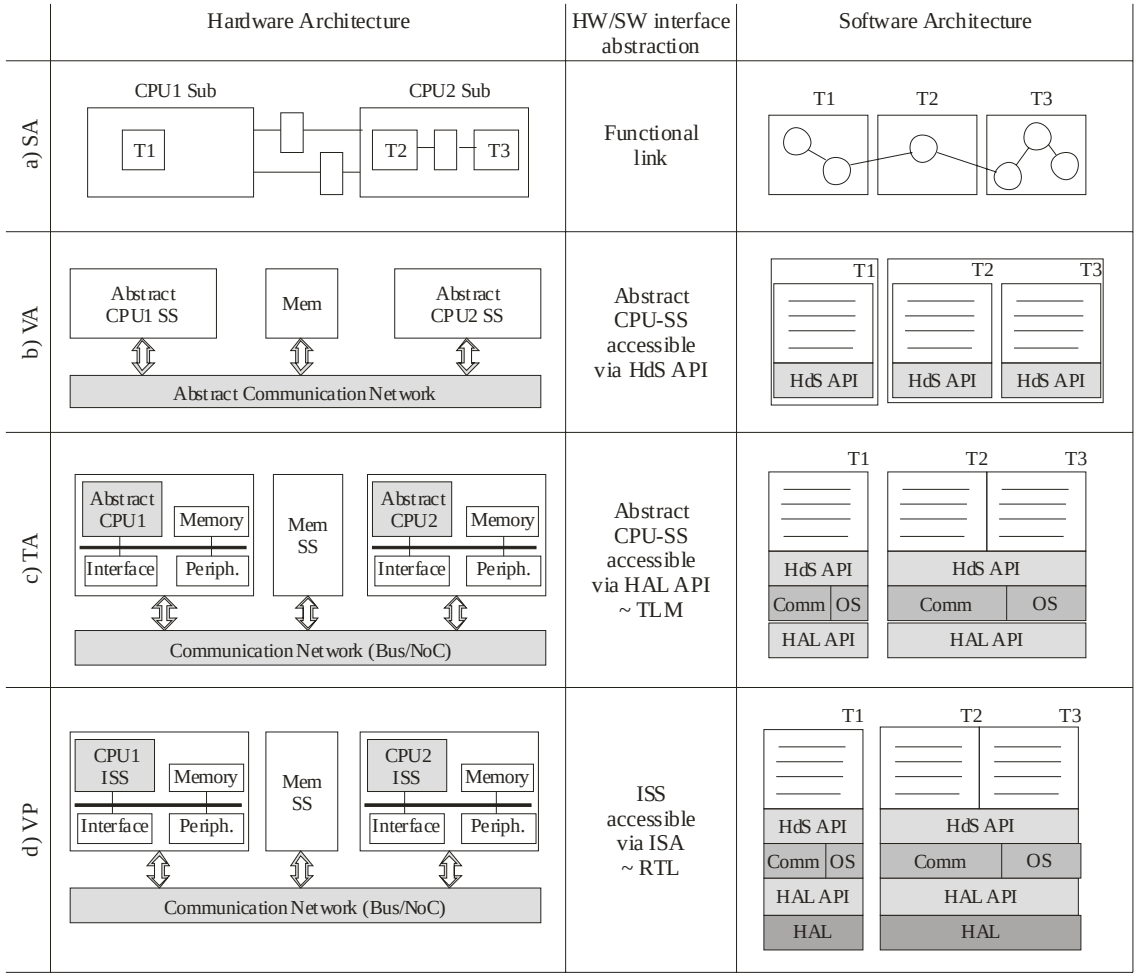
Piecing it all together …

- Compiler
  - Combines
    - Functional design
    - Architecture
  - Generates
    - Hardware (HDL)
    - Software (C/ASM)
- Simulator to
  - Explore design space
  - Verify design choices
    - Hardware in the loop
    - Processor in the loop
    - Silicon in the loop
    - …



Implementation effects

- Application algorithm validation
- Task code validation
  Deadlock free execution
- Partitioning validation
- Communication architecture
  Number of exchanged bytes
- HdS Integration validation
- Communication architecture
  Number of exchanged bytes
  Number of conflicts on bus
  Level of NoC congestion
- Final SW binary validation
- Performance measurement
- Memory mapping validation



Acknowledgment: *picture redrawn from work by Katalin Popovici and Ahmed Amine Jerraya*

Research

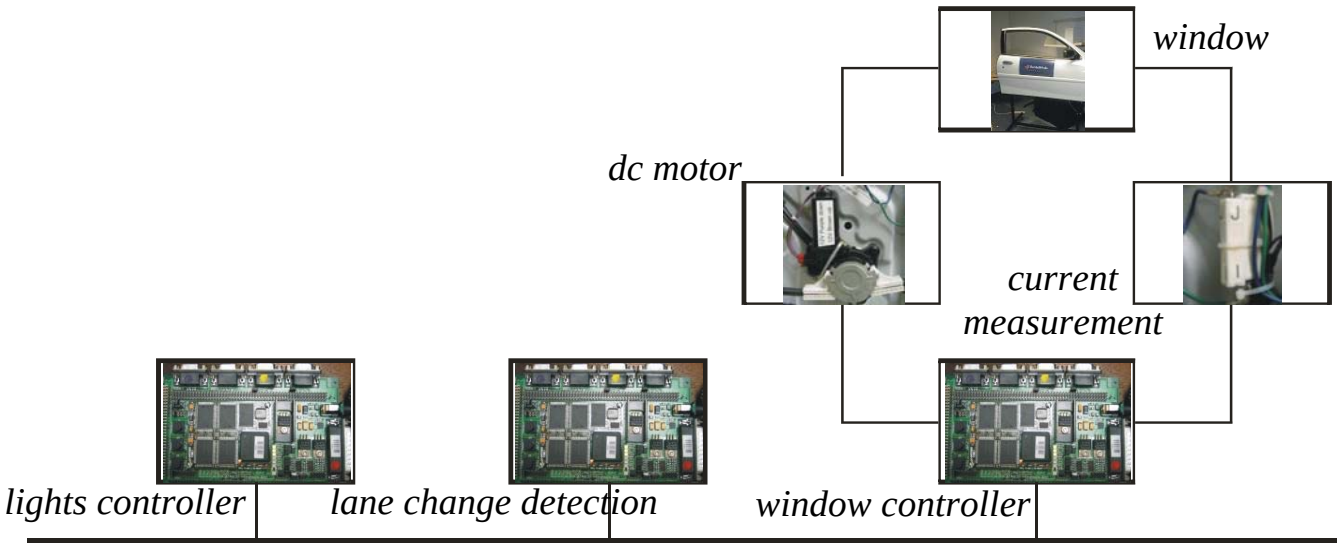
- Include performance metrics
- Systems of systems
  - Compositionality
  - Composability
- How to deal with heterogeneity?
  - Dieting philosophers
- Can you prove abstractions are property preserving?

Agenda

- Model-Based Design
- Computer Automated Multiparadigm Modeling
- Application
- Multi-formalism modeling
- Mixed-signal simulation
- Hybrid Dynamic Systems
- Summary

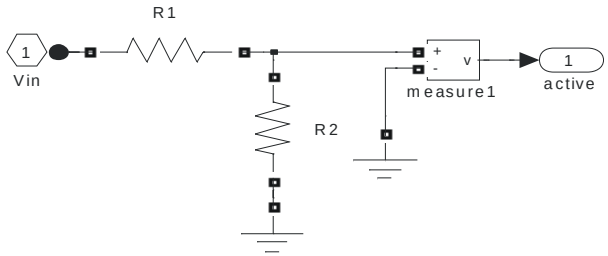
Networked Embedded Control System

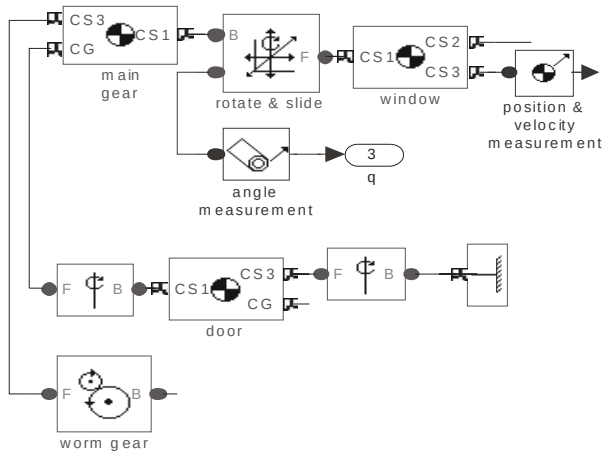
- Continuous and discrete behavior
  - Plant; Continuous-time behavior with sporadic discrete events
  - Controller; Frequent periodic events
  - Network; Frequent aperiodic events



Modeling the plant physics

- Domain-specific modeling
  - DC motor and signal conditioning—Power Systems
  - Window lift mechanism—Multibody Systems

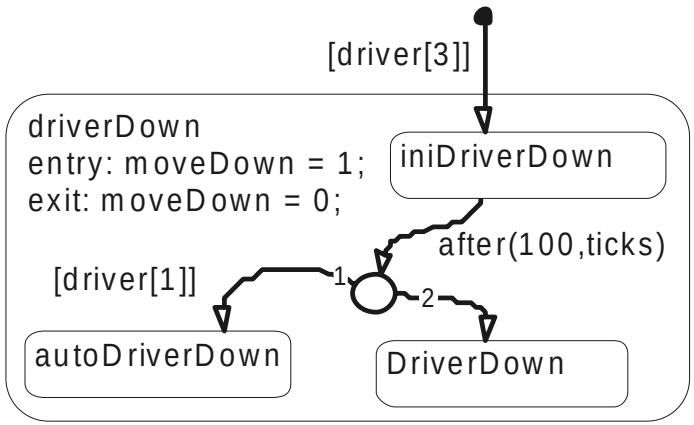




- Characteristics
  - **Noncausal**, energy-based, modeling
  - **Differential equation** based

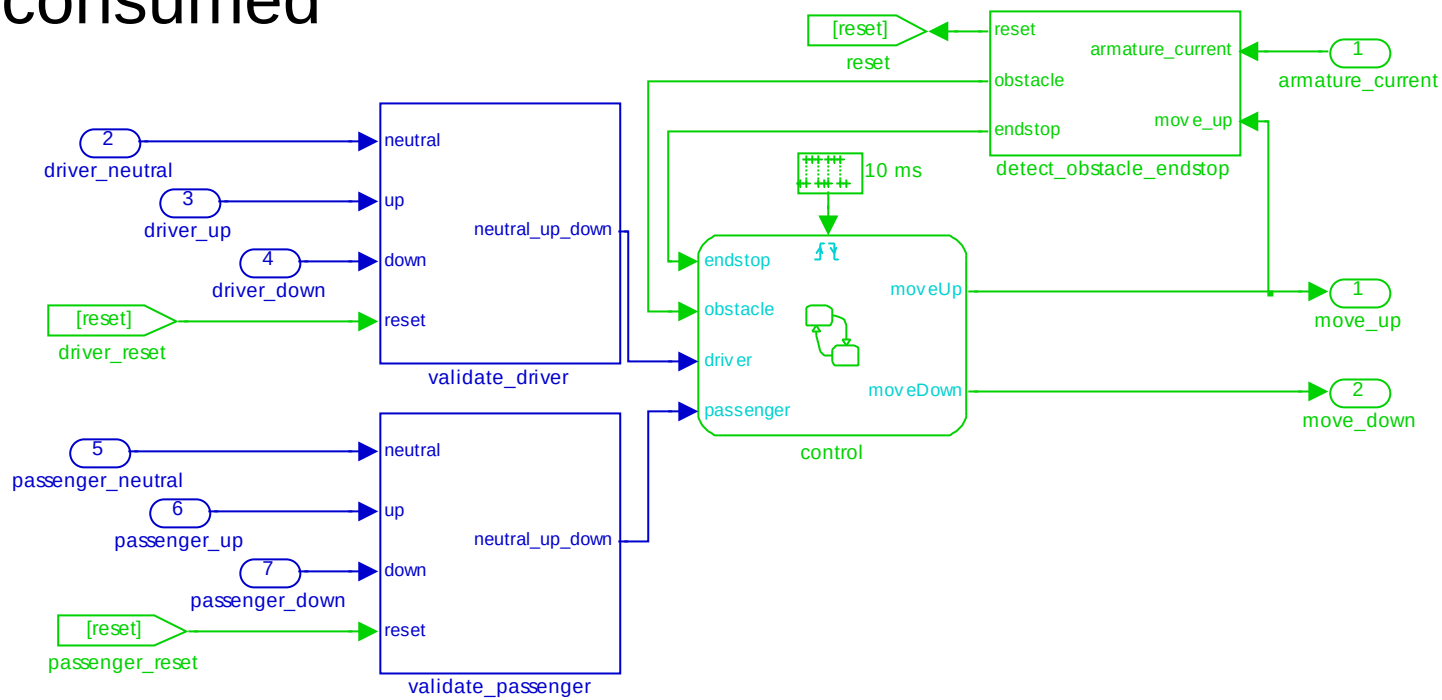
Modeling the supervisory control

- Discrete state based
- Discrete events cause transitions between states
- Conditions to guard the transition
- **Untimed**
- **Control centric**, **consume**



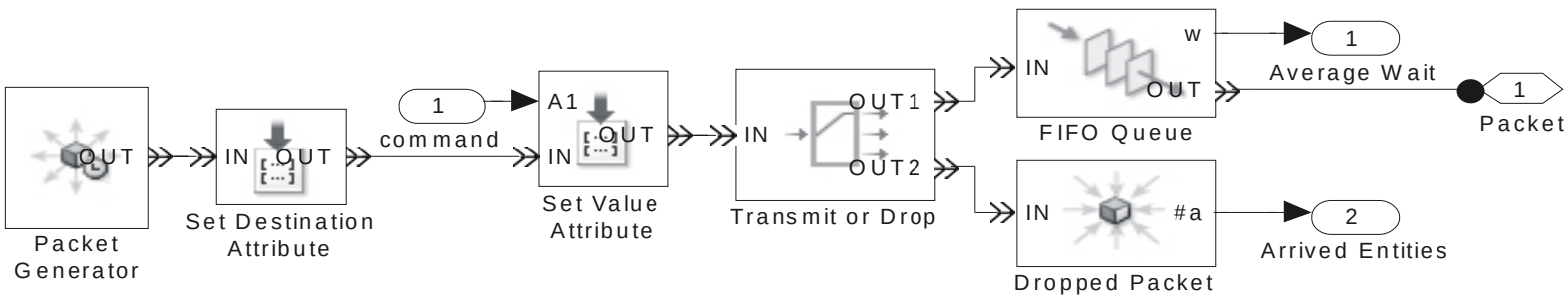
Modeling the feedback control

- Embedded processors at fixed sample rate
- Sampled discrete time
- **Periodic**, **infinite read**; data values are not 'consumed'

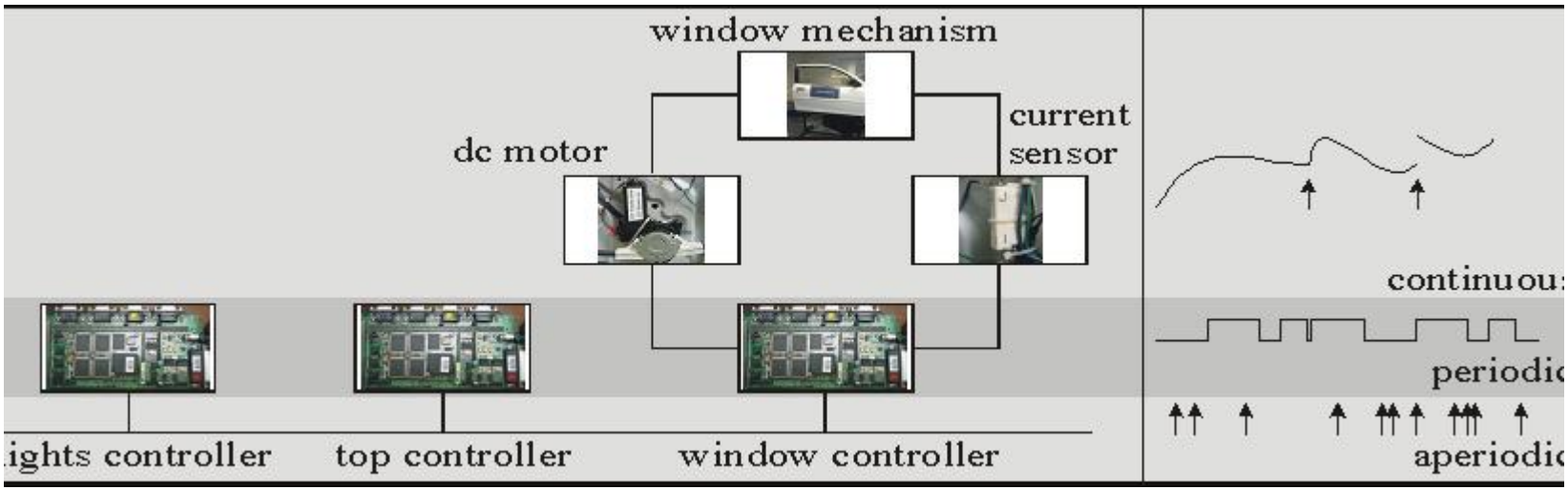


Modeling network traffic

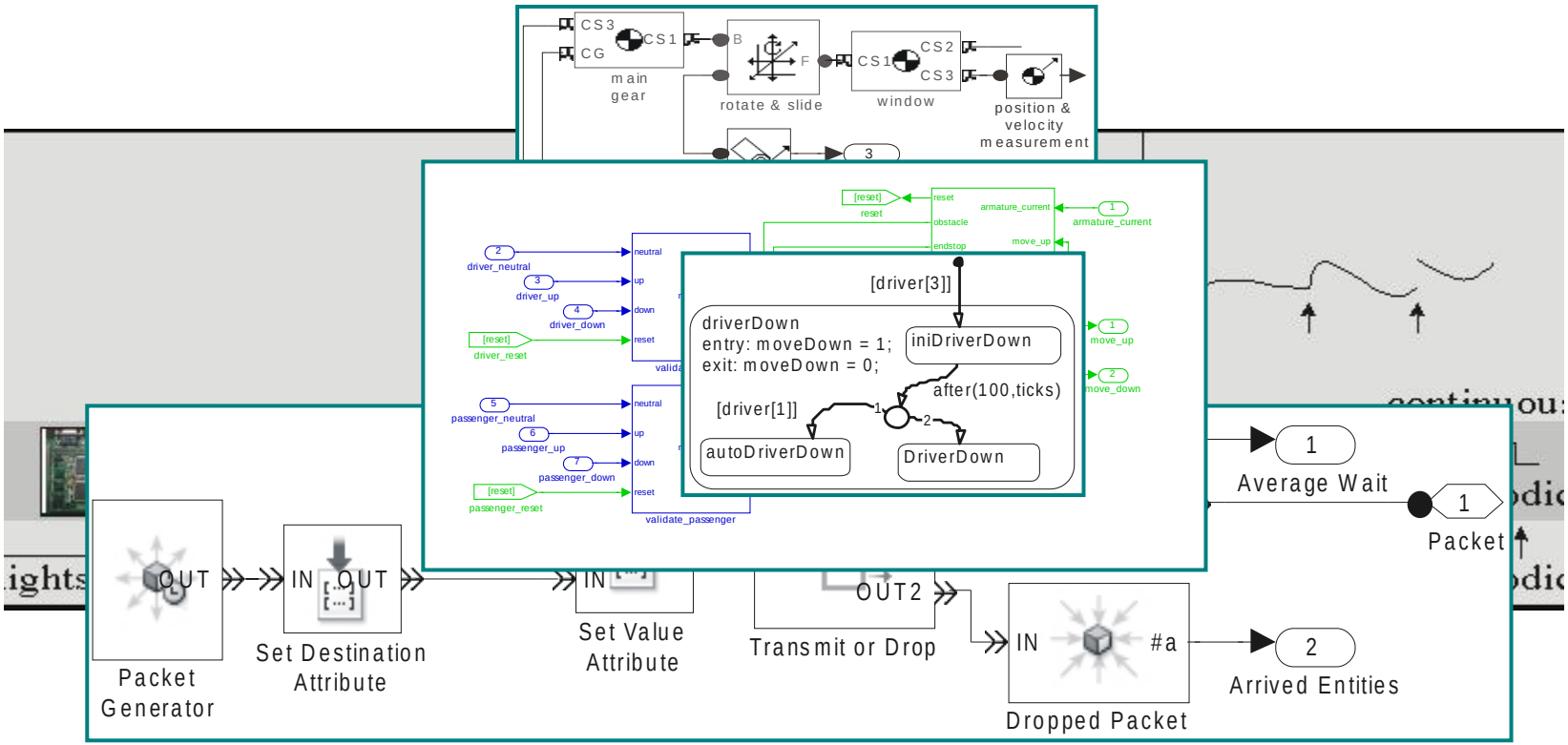
- Packets moving from one point to another
  - Attributes
    - Source, destination, service time, priority
  - Preemption
- Server/Queue systems
  - Entities flowing through a graph, **data centric, producer/consumer**
  - Discrete events, **aperiodic**, often *stochastic*



All part of the networked embedded system



All part of the networked embedded system



Research

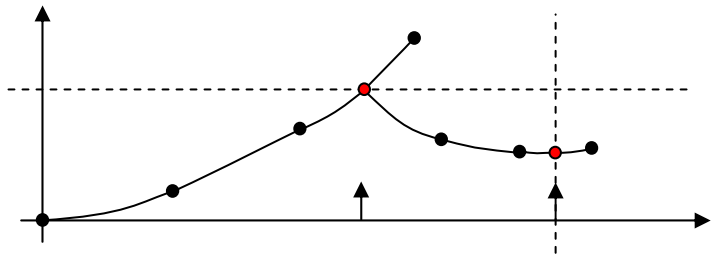
- How to handle multiple formalisms
  - Relate
  - Combine
  - Integrate
- Formalisms evolve
  - Forward and backward compatibility

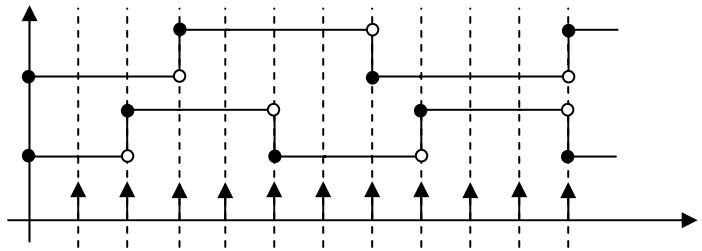
Agenda

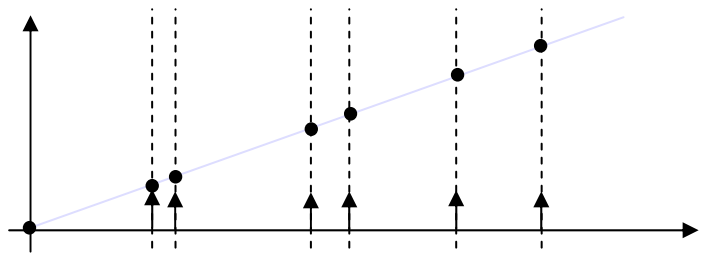
- Model-Based Design
- Computer Automated Multiparadigm Modeling
- Application
- Multi-formalism modeling
- Mixed-signal simulation
- Hybrid Dynamic Systems
- Summary

How to handle a discrete event

- Time-driven
  - Time integrated
    - Integrate up till event time
    - Inefficient for time events
  - Sampled time
    - Run scheduler at lowest rate
    - Inefficient for widely spaced events
- Event-driven
  - Jump to event time immediately
  - Does not apply to state events







Classes of behavior

- Plant, time integrated (**TI**)
  - Continuous time
  - Sporadic aperiodic *state-events* (zero crossings)
  - Events occur during execution
- Controller, sampled time (**ST**)
  - Discrete time
  - Frequent periodic *time-events*
  - Events are known before execution
- Network/OS, event driven (**ED**)
  - Discrete time
  - Frequent aperiodic *time-events*
  - Events occur during execution

Event classification

	predict	unknown
periodic	**ST**	-
aperiodic	**ED**	**TI**

Dedicated Time-Driven Solver

- Numerical solver integrates time (step *h*)

$$k_1 = h \cdot f(x_n, y_n)$$

$$k_2 = h \cdot f(x_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$

$$k_3 = h \cdot f(x_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$

$$k_4 = h \cdot f(x_n + h, y_n + k_3)$$

$$y_{n+1} = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5)$$

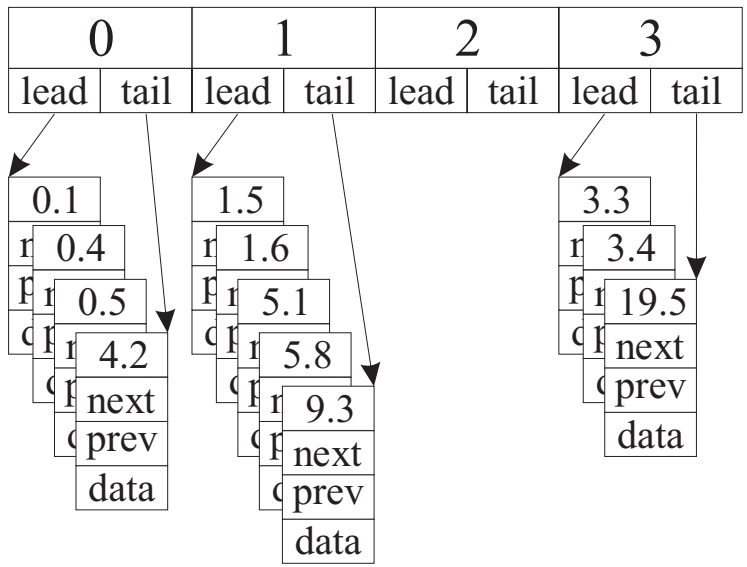
- Sampled time event schedule (time up till which to integrate)

Dedicated Event-Driven Solver

- Event-driven solver
  - Event calendar

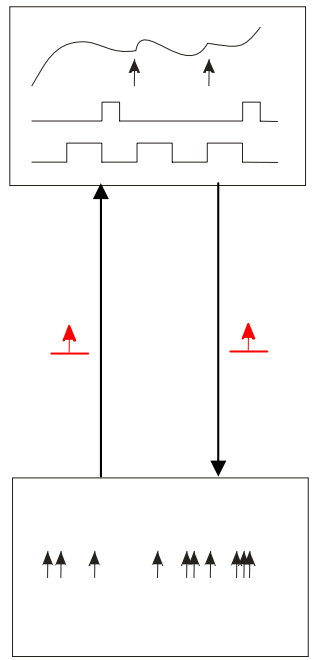
20 [ms]	`open_cover`
2020 [ms]	`move_top_up_cmd`
2150 [ms]	`move_down_window`
2250 [ms]	`stop_moving_window`

  - Efficient data structure

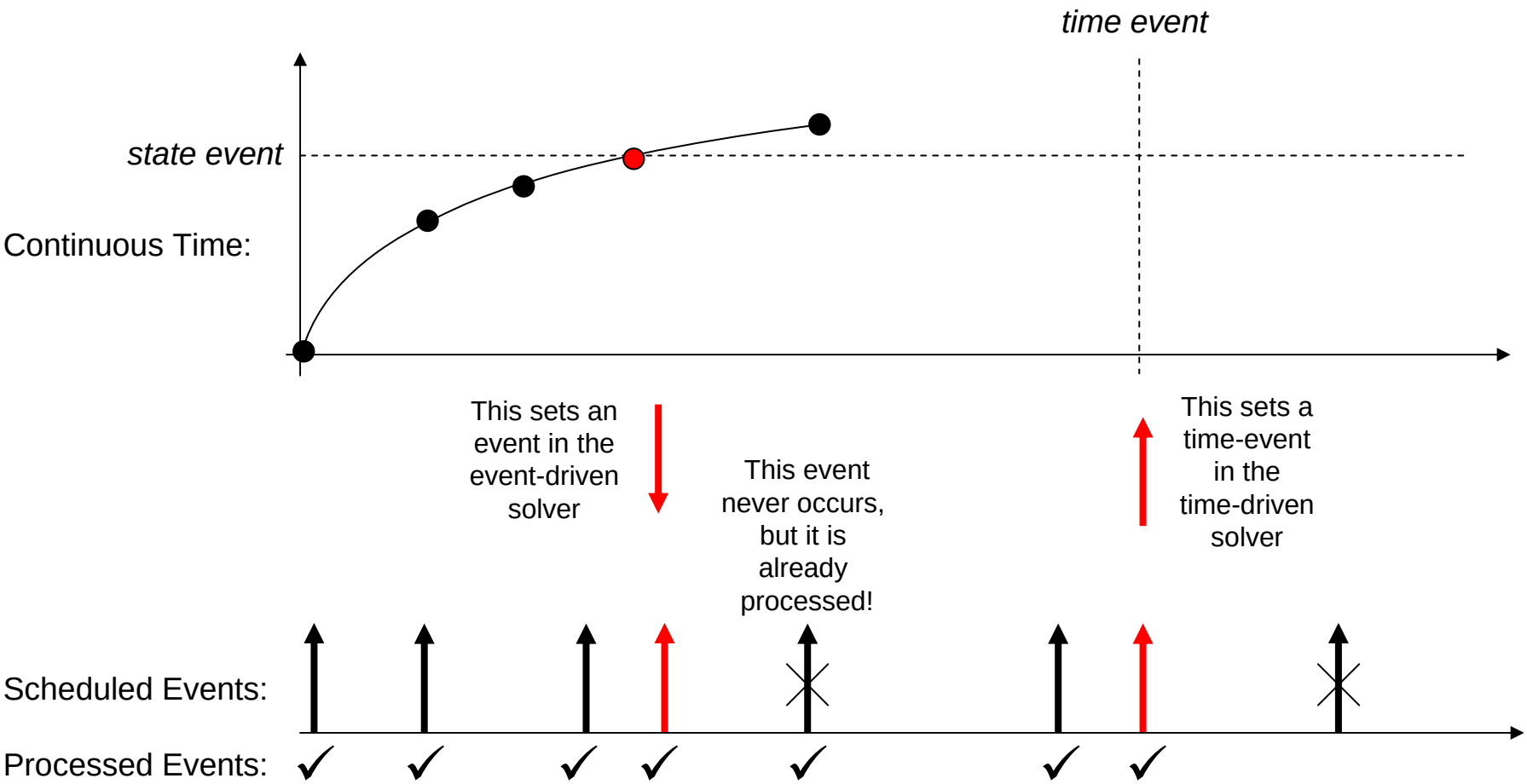


Efficiency

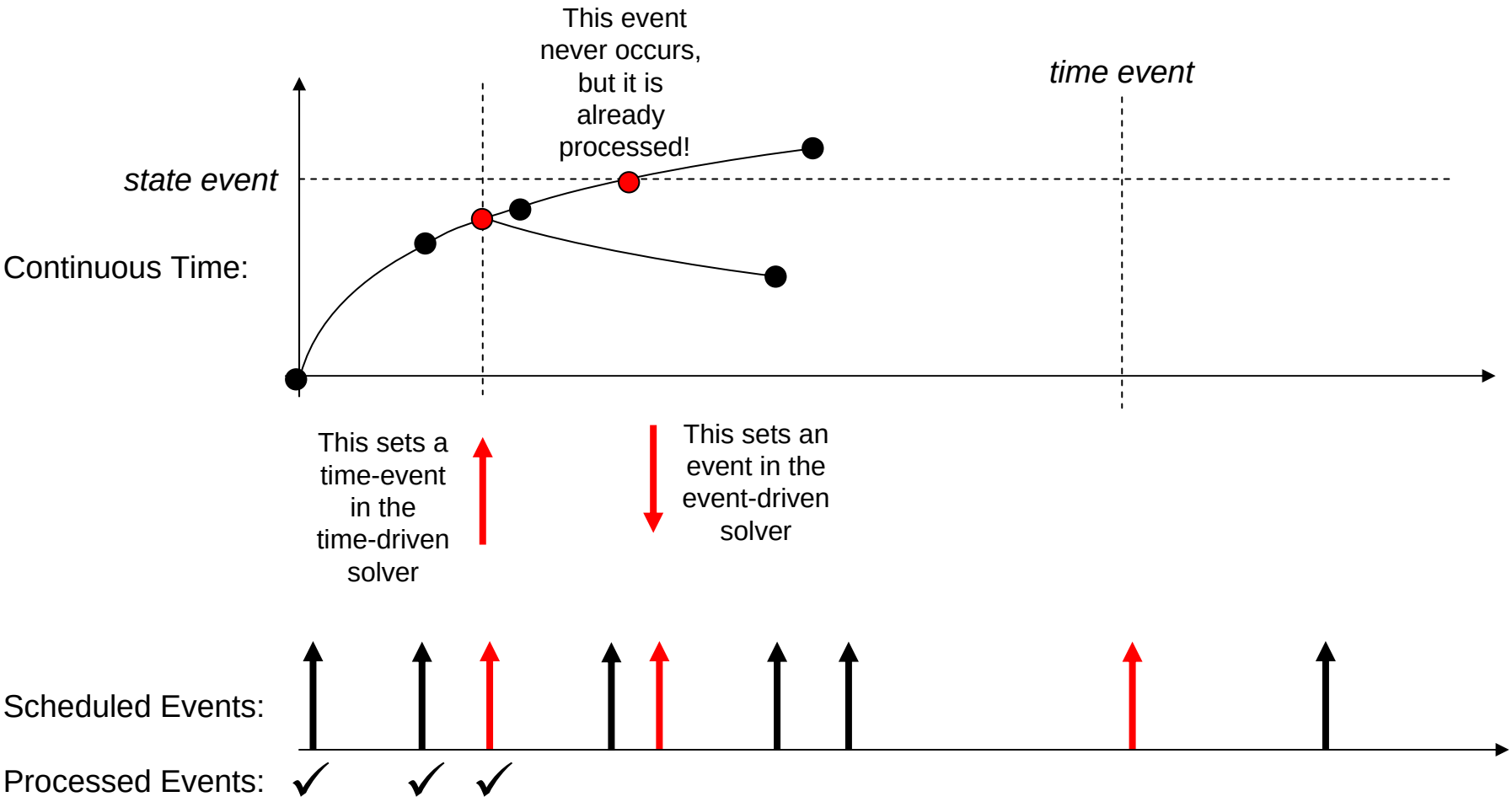
- Two separate solvers
  - Time-driven solver
    - Numerical solver integrates time (step *h*)
    - Sampled time event schedule (time up to which to integrate)
  - Event-driven solver
    - Event calendar
- Challenges
  - Combine the solvers
  - Integrate the solvers



Event-driven leads



Time-driven leads



Questions

- Can we restrict the modeling constructs that are allowed?
  - For example, no state events allowed to trigger events in the event-driven solver
- What inaccuracy is acceptable?
  - Not do zero-crossing detection
  - Run event-driven part at a quick base rate
- What efficiency is necessary?
  - Lock step approach
  - Use one solver all together?
    - Do you lose the ability to handle a batch of discrete events independently?
  - Other techniques?
    - Model differently
- What is the preferred configuration?
  - Time-driven leads, store continuous-time state
  - Event-driven leads, store discrete event state
  - Alternatives?

Agenda

- Model-Based Design
- Computer Automated Multiparadigm Modeling
- Application
- Multi-formalism modeling
- Mixed-signal simulation
- Hybrid Dynamic Systems
- Summary

What is a hybrid dynamic system?

- Combines two types of behavior
  - Continuous-time
    - Ordinary differential equations (ODEs)

$$f_{\alpha_i} : \dot{x} = A_{\alpha_i} x + B_{\alpha_i} u$$

    - Differential and algebraic equations (DAEs)

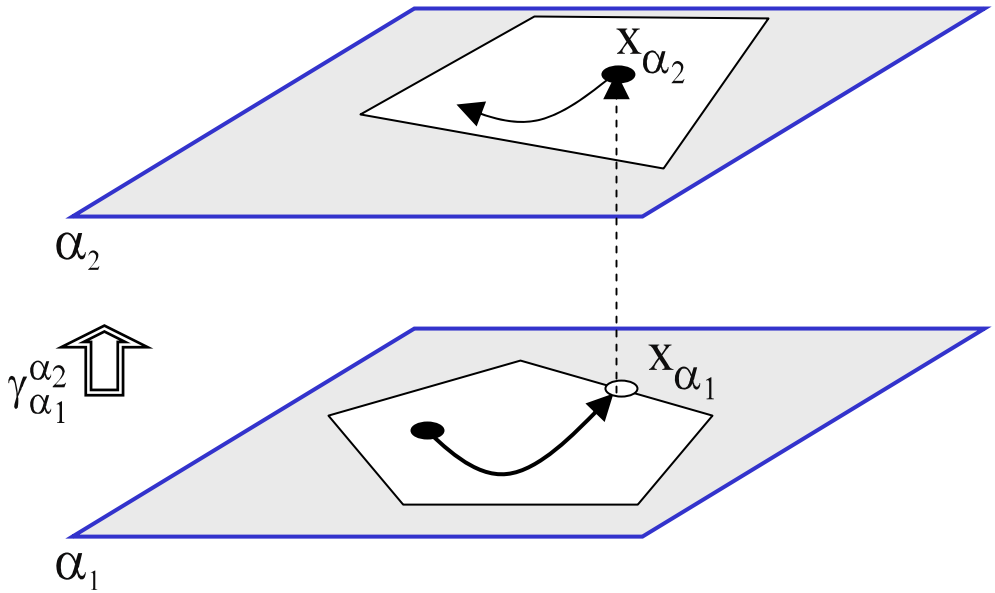
$$f_{\alpha_i} : E_{\alpha_i} \dot{x} = A_{\alpha_i} x + B_{\alpha_i} u$$

  - Discrete state changes
    - Inequalities

$$\gamma_{\alpha_i}^{\alpha_{i+1}} : C_{\alpha_i} x + D_{\alpha_i} u \geq 0$$

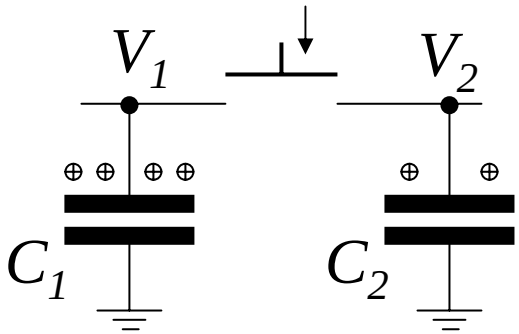
Hybrid Dynamic Behavior

- Geometric view
  - Modes of continuous, smooth, behavior
  - Patches of admissible state variable values



Generalized state space

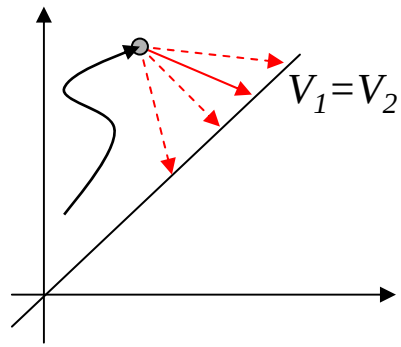
- Dimensions of state space may collapse



$$V_i = \frac{q_i}{C_i}$$

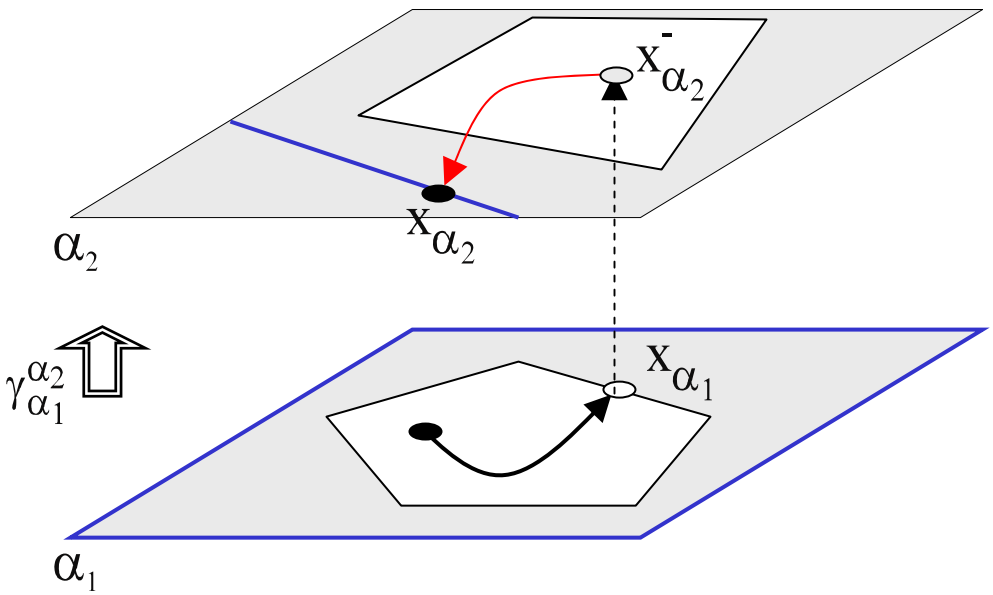
$$I = C_i \frac{dV_i}{dt}$$

- After switch closes
  - $V_1 = V_2$
  - Instantaneous change in charge



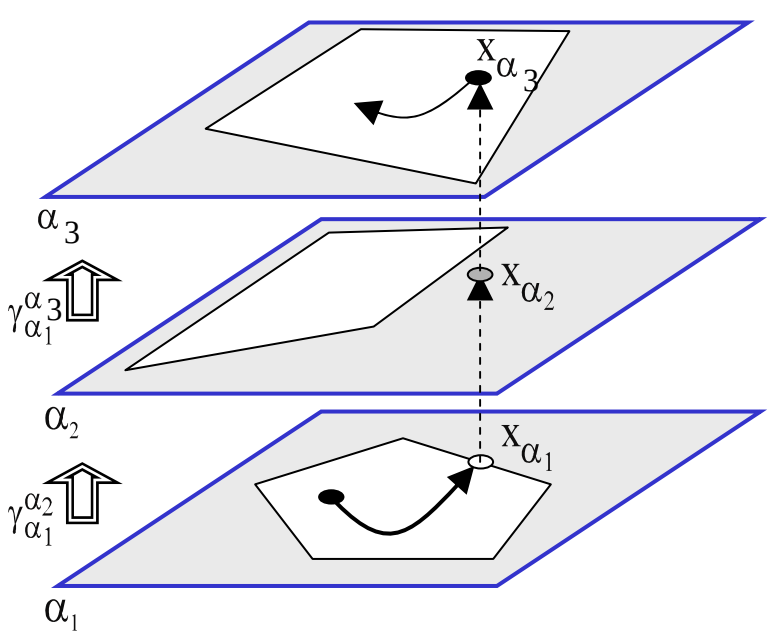
Refined Hybrid Dynamic Behavior

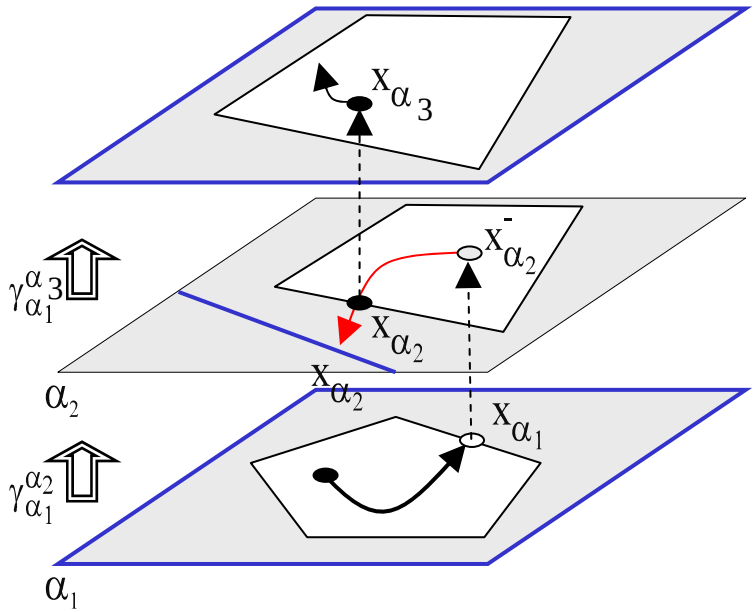
- Geometric view
  - Modes of continuous, smooth, behavior
  - Patches of admissible state variable values
  - Manifold of dynamic behavior



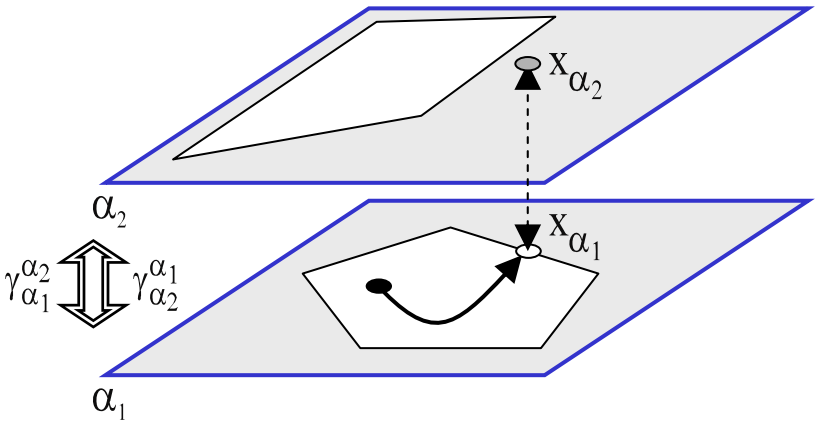
Sequences of mode changes

- Mythical mode
- Pinnacle

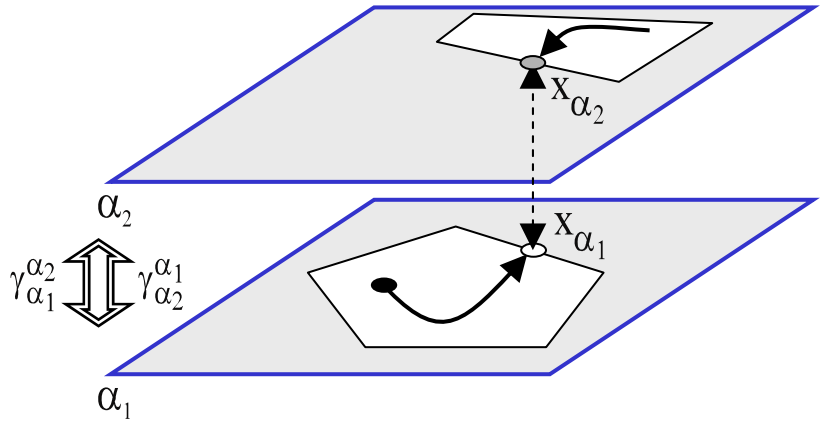




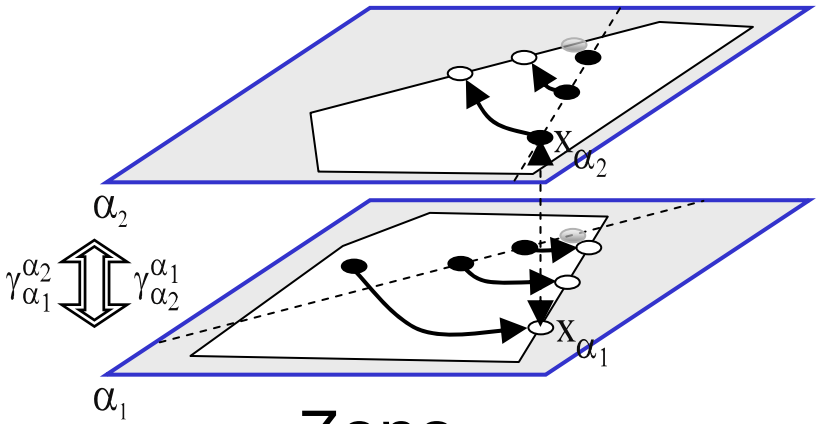
Pathological behaviors



Divergence of time



Chattering



Zeno

Ontology

- Phase Space Transition Behavior Classification
  - Mythical (state invariant)
  - Pinnacle (state projection aborted)
  - Continuous
    - Interior (continuous behavior)
    - Boundary (further transition after infinitesimal time advance)
    - Sliding (repeated transitions after each infinitesimal time advance)
- Combinations of Behavior Classes
- On Zeno
  - *Chattering*: infinitesimal time advance
  - *Divergence of time*: infinitely many switches without time advance
  - *Zeno*: infinitely many switches never past a point in time

Research

- How to efficiently generate behavior for combined time-driven and event-driven execution engines?
- How to deal with run-time index changes?
- Pathological behaviors
  - How to detect in industrial-size models?
  - How to (re)solve?

Agenda

- Model-Based Design
- Computer Automated Multiparadigm Modeling
- Application
- Multi-formalism modeling
- Mixed-signal simulation
- Hybrid Dynamic Systems
- Summary

Summary

- Model-Based Design
  - Efficient engineering system design
- Important research
  - Computer Automated Multiparadigm Modeling
  - Hybrid Dynamic Systems
  - ...
- Multi-domain research
  - Electrical engineering
  - Computer engineering
  - Theoretical physics
  - Computer science

Acknowledgments

- Hans Vangheluwe
- Ben Denckla
- Katalin Popovici
- Mirko Conrad
- All participants of the annual Computer Automated Multiparadigm Modeling (CAMPaM) workshop at the McGill University Bellairs Campus, Barbados
- …