HW/SW Cyber-System Modeling Tools

Julio Oliveira

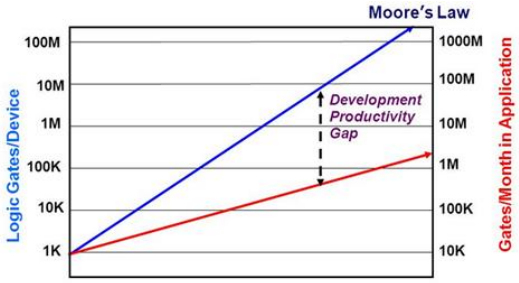
Karol Desnos

HW/SW Cyber-System Modeling Tools

1. Modeling Environment and Languages
2. (HW &) SW Synthesis
3. Simulators
4. Verification / Validation
5. Runtime Support

From System Models to Value

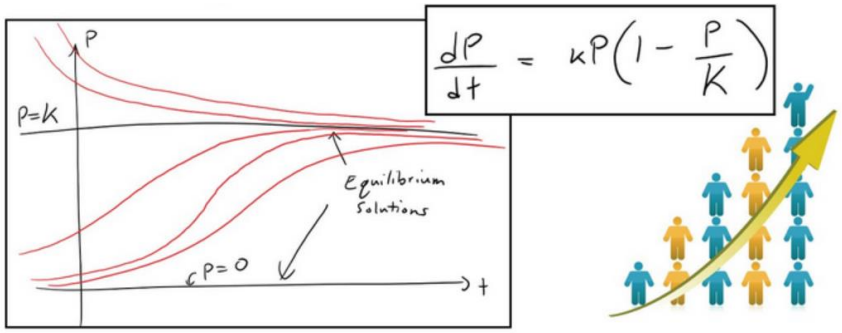
The topic of this lecture



- Communicate
- Validate - Verify
- Configure
- Analyze
- Simulate
- Decide
- Optimize
- Automation !

A short recap from CPS Modeling lecture

We talked about models …



… and about modeling for CPS.

Abstraction (Simplification)

Description (Specification)

Operational (Executable)

A short recap from CPS Modeling lecture

We talked about multi-aspect modeling…

UML MARTE

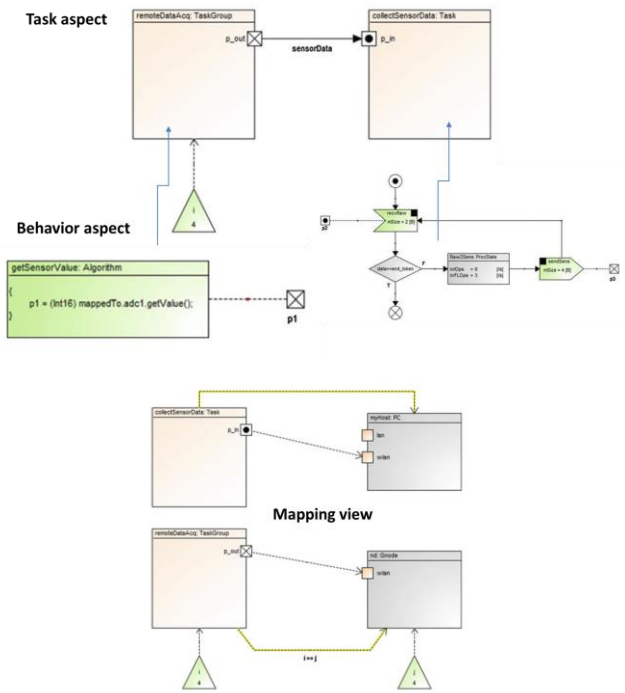
OMG SYSTEMS MODELING LANGUAGE SysML™

AADL

ISO IEC ITU RM-ODP

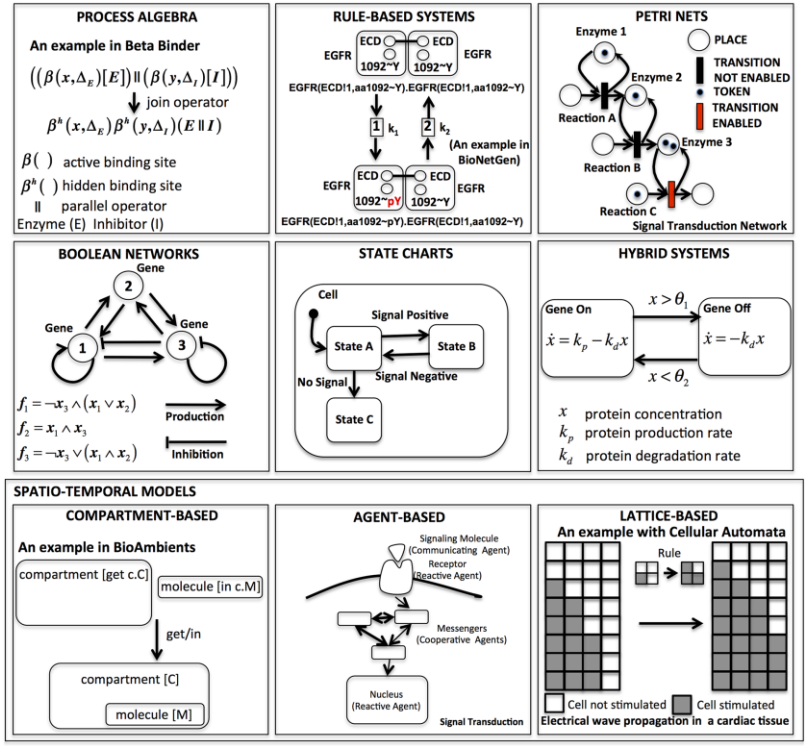
UNIFIED MODELING LANGUAGE UML™

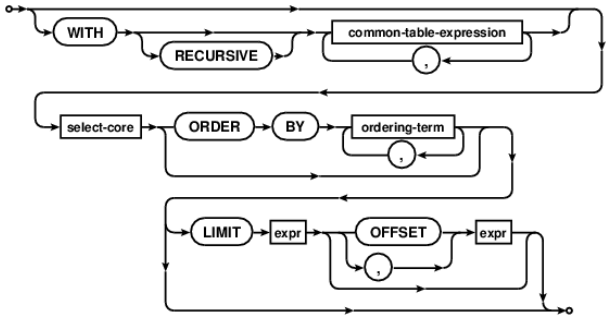
… and illustrated how aspects can be combined to extract KPIs.

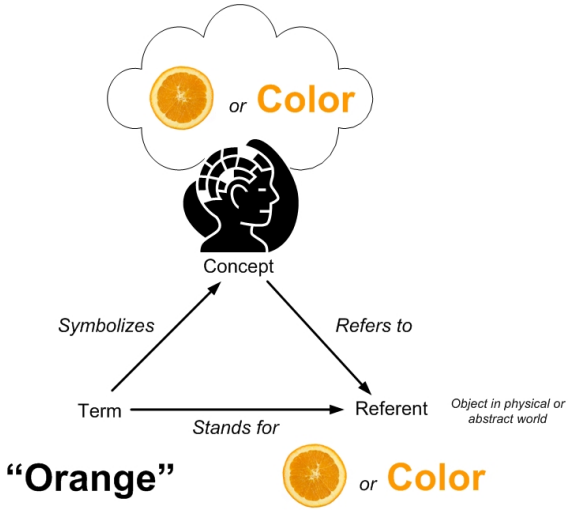


Last, but not least…

We went into details about Semantics, Syntax, Models of Computation and Models of Architecture

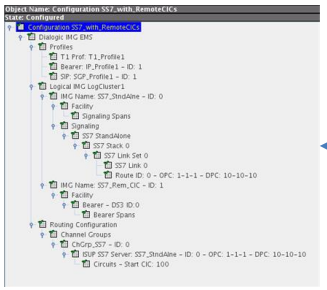


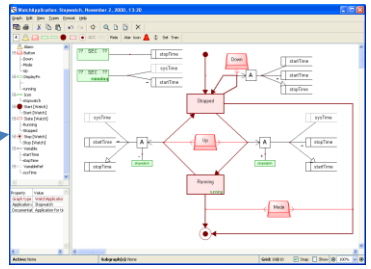




Making more of a model

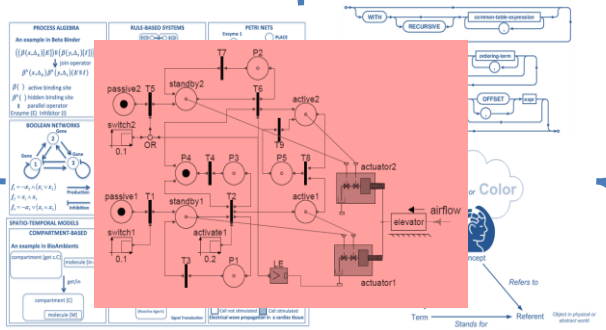
Configuration

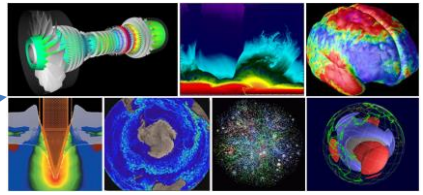




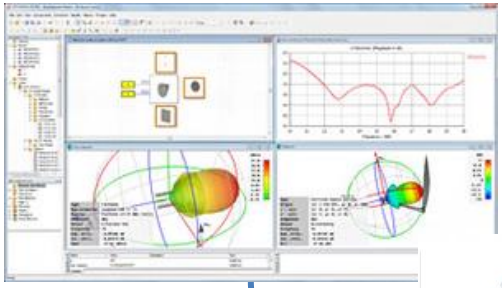
Modeling environment

Analysis
Validation / Verification
Simulation

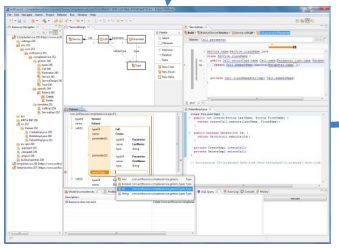




Visualization
Presentation



Automatic design
Design space exploration
Decision making

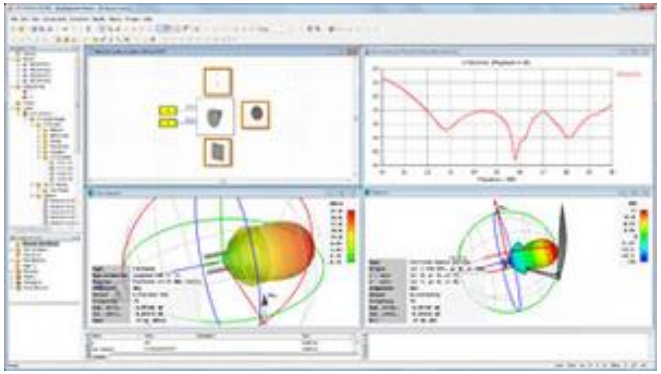


Code synthesis

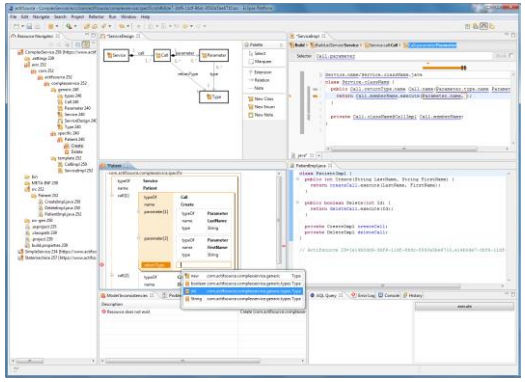
Implementation

In this lecture

Analysis
Validation / Verification
Simulation



Code synthesis



Compilers for adaptive systems

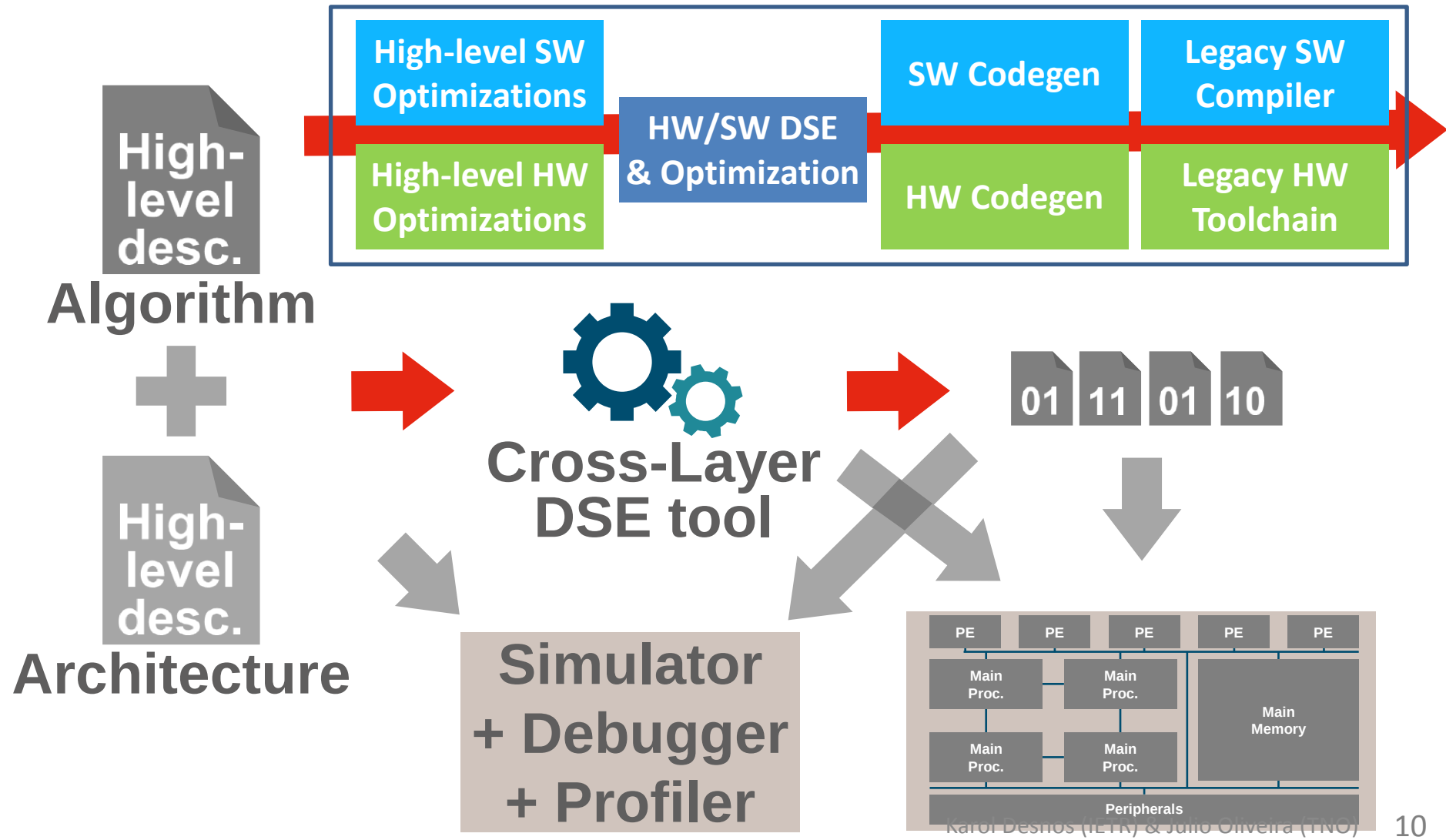
HW/SW Cyber-System Modeling Tools

1. Modeling Environment and Languages
2. (HW &) SW Synthesis
3. Simulators
4. Verification / Validation
5. Runtime Support

Source: Bhattacharyya, S. S., Murthy, P. K., & Lee, E. A. (2012). *Software synthesis from dataflow graphs* (Vol. 360). Springer Science & Business Media.

(HW &) SW synthesis

Synthesis at component level?

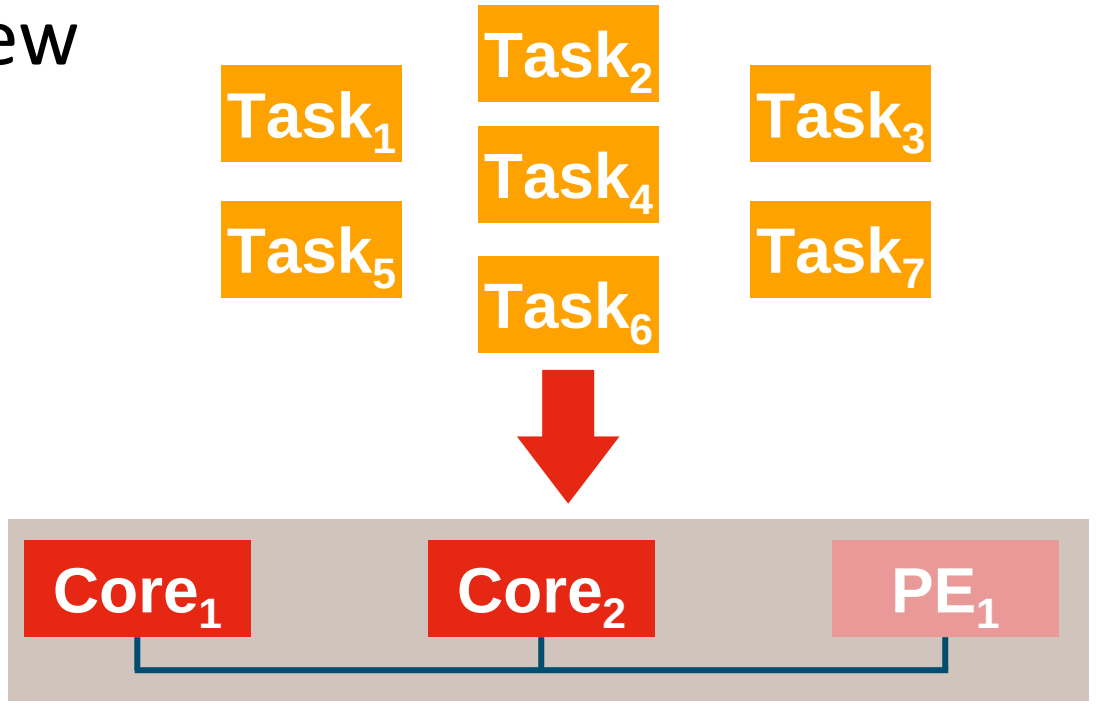


(HW &) SW synthesis

SW synthesis > overview

Task Management

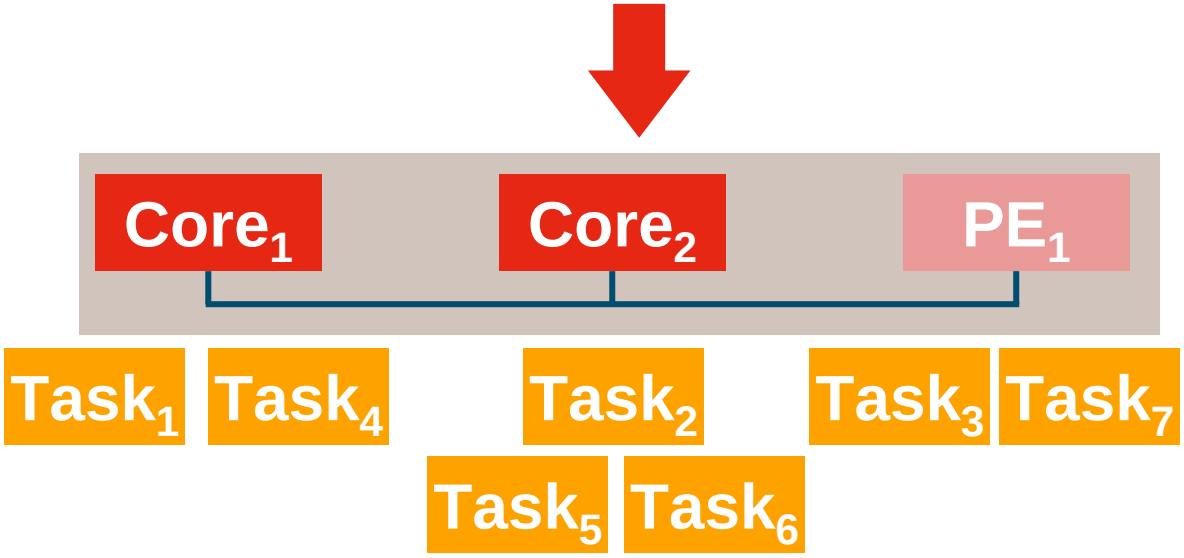
- Assignment (Mapping)
- Ordering (Scheduling)
- Timing



SW synthesis > overview

Task Management

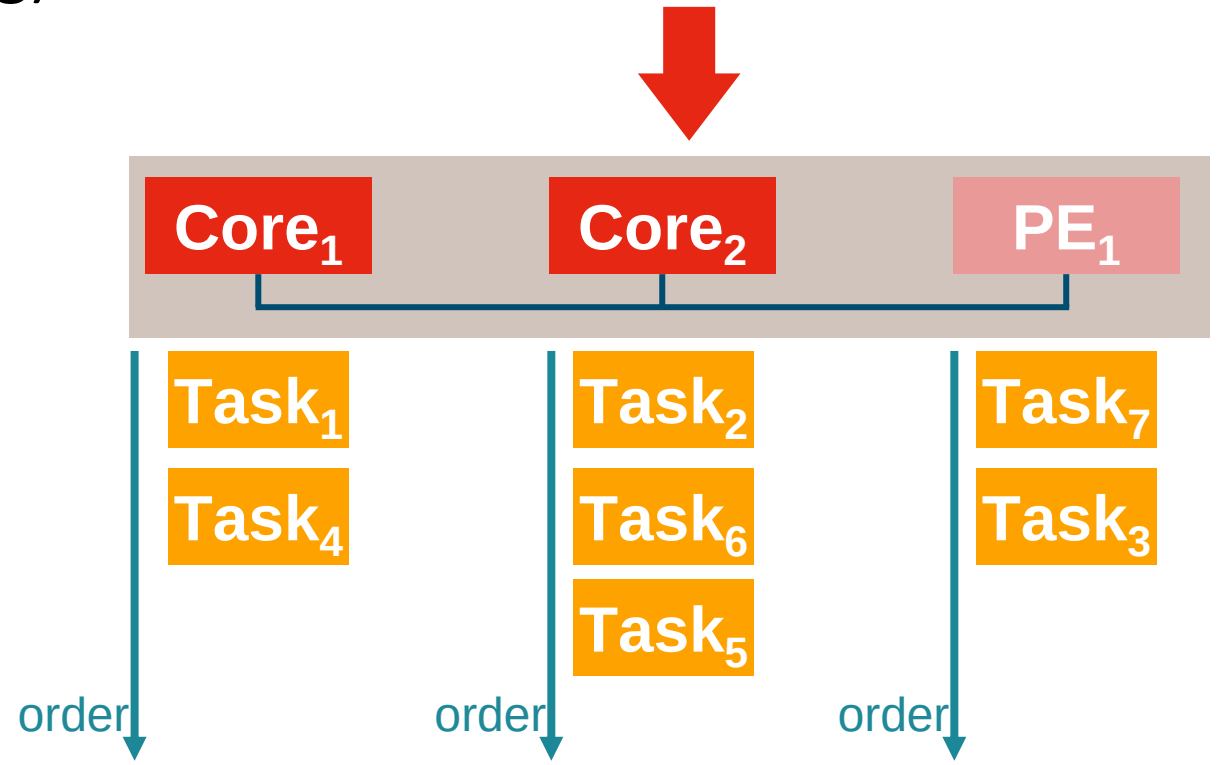
- Assignment (Mapping)
- Ordering (Scheduling)
- Timing



SW synthesis > overview

Task Management

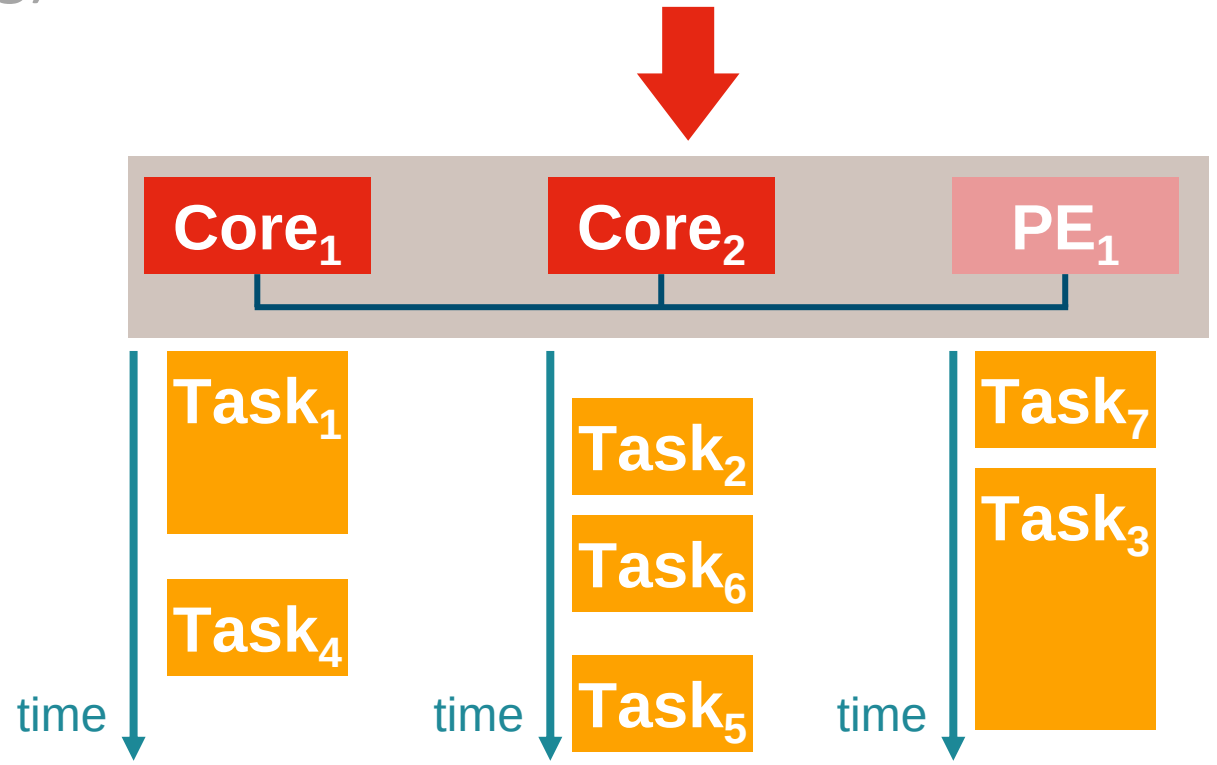
- Assignment (Mapping)
- Ordering (Scheduling)
- Timing



SW synthesis > overview

Task Management

- Assignment (Mapping)
- Ordering (Scheduling)
- Timing



(HW &) SW synthesis

SW synthesis > HW management

Task Management

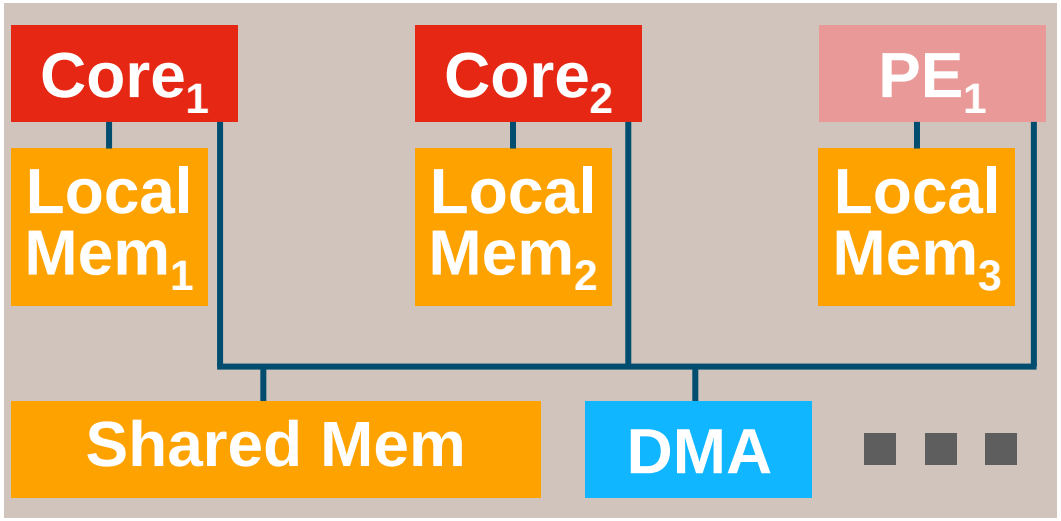
- Assignment (Mapping)
- Ordering (Scheduling)
- Timing

Memory

- Static Allocation
- Dynamic Allocation
- Model-based allocation

Communication

- Direct copy
- DMA
- Ordering



Synchronization

- Polling
- Interrupts

(HW &) SW synthesis

SW synthesis > Complexity

HW resources heterogeneity

- Processing Elements / Interconnect / Memory

Limited HW resources

- PE / Interconnect / Memory

SW model complexity

- Complex control dependency
- Complex data dependency
- Low predictability

(HW &) SW synthesis

SW synthesis > Tools & methods objectives

Optimizing / offering trade-offs between:

- Latency / Response time
- Throughput
- Load balancing
- Memory footprint
- Power consumption

Adaptability to any target architecture:

- DSP
- GPU
- HPC

Mapping/Scheduling > Problem

Part of "Operational Research"

- How to organize a production line, train schedule, …
- How to organize a project (Gantt Chart, …)
- How to make decisions in general

NP-Complete Problem

- Validity of a solution to the problem can be verified in polynomial time (e.g. verifying that a schedule is valid).
- No polynomial time algorithm for solving NP-complete problems is known (and it is likely that none exists).
- When the problem grows (e.g. number of cores or tasks), solving it is becoming more complex exponentially.

Mapping/Scheduling > Solutions

Heuristic algorithms

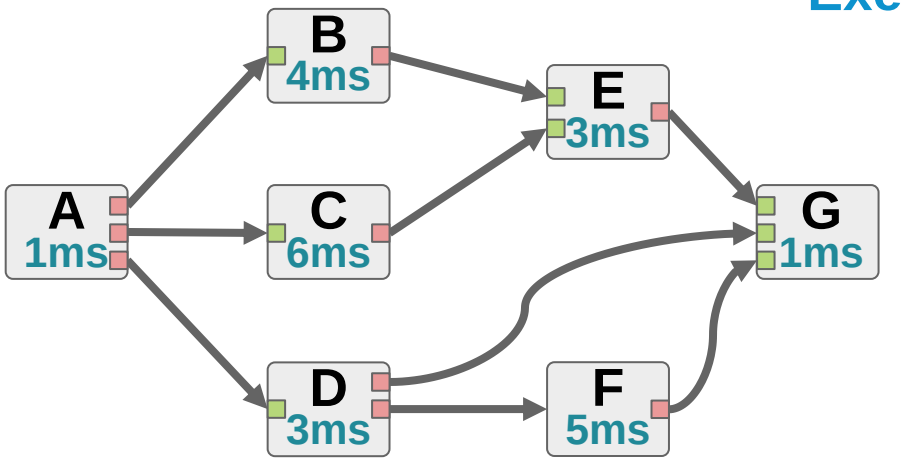
- Find a sub-optimal solution in polynomial time
- No guarantee on the solution quality

	Problem Specific	Generic Algorithms
Constructive	• List scheduling • Greedy scheduling • Hybrid flow-shop	• Divide and conquer • Branch and bound
Iterative	• FAST scheduling	• Integer Linear Programming • Genetic Algorithms • Simulated annealing • Ant colony

Source: Z. Peng, lecture notes of "Computer aided design of electronics", LiU

Mapping/Scheduling > List scheduling for dataflow

1. Create a list of actors sorted in:
   - Topological order (i.e. data dependency order)
   - When equivalent, secondary sorting criteria is used: longest execution time, critical path before last task, …



Longest Execution time

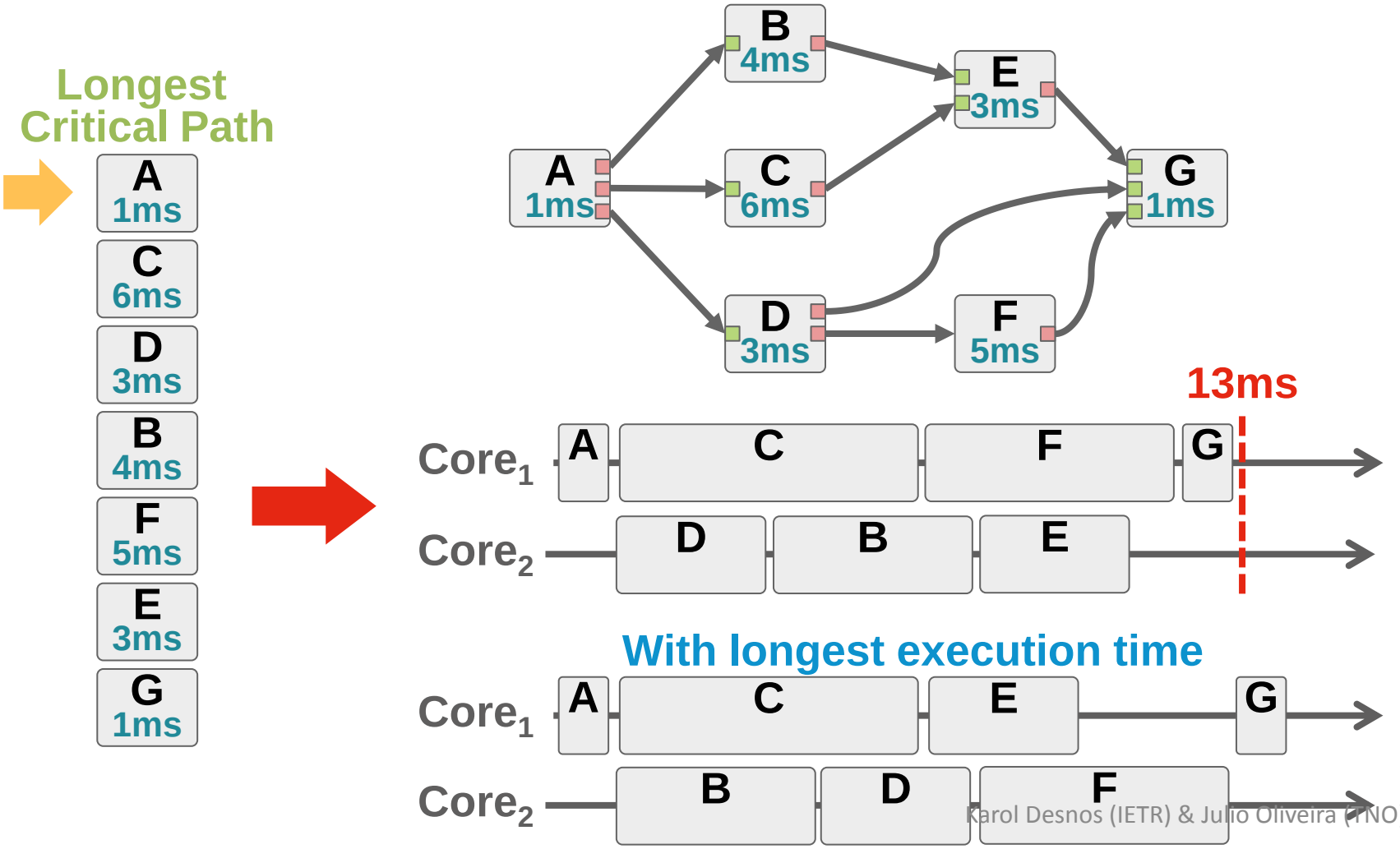
A 1ms
C 6ms
B 4ms
E 3ms
D 3ms
F 5ms
G 1ms

Longest Critical Path

A 1ms
C 6ms
D 3ms
B 4ms
F 5ms
E 3ms
G 1ms

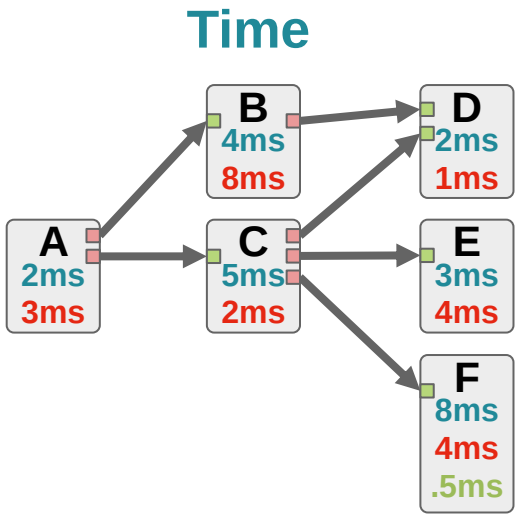
Mapping/Scheduling > List scheduling for dataflow

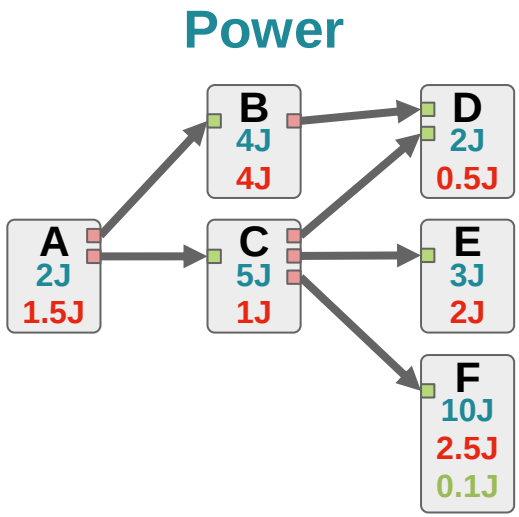
2. Map and schedule actors to the first available PE:

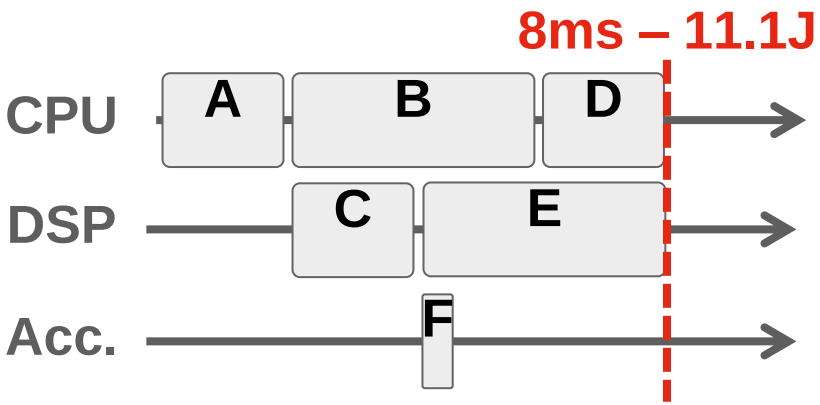


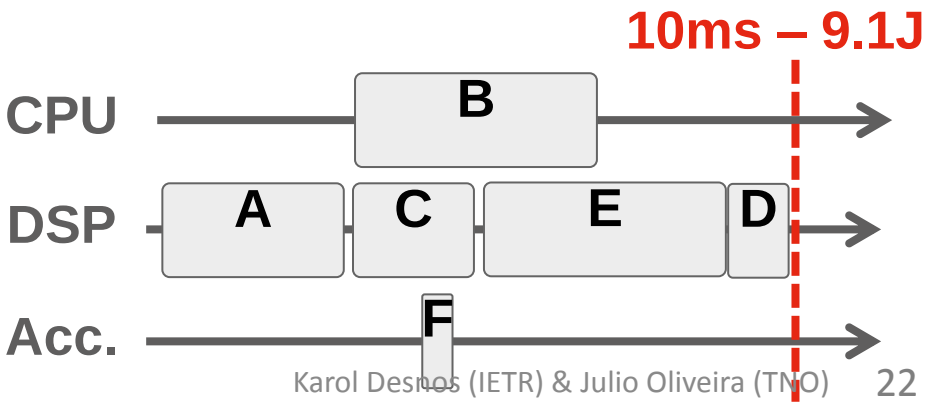
Mapping/Scheduling > Multi-objective optimization

E.g. Latency and power on heterogeneous target.

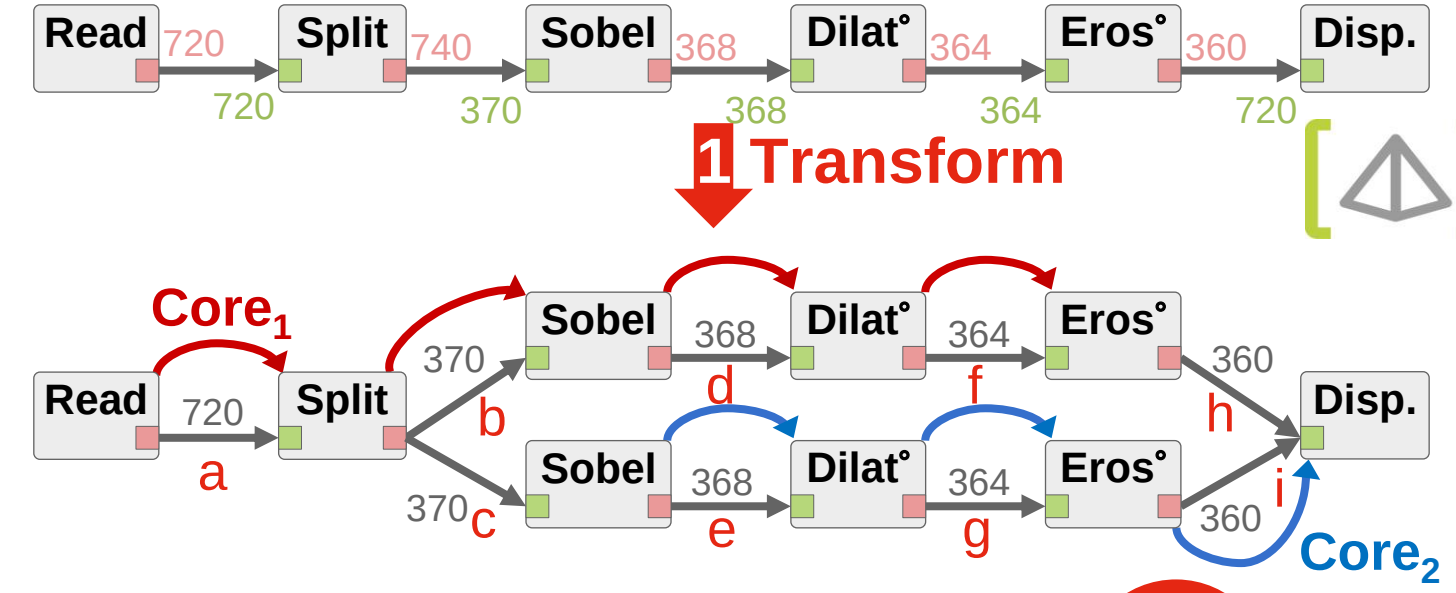








Memory Allocation



Memory Exclusion Graph (MEG)

HW/SW Cyber-System Modeling Tools

1. Modeling Environment and Languages
2. (HW &) SW Synthesis
3. Simulators
4. Verification / Validation
5. Runtime Support

Simulators

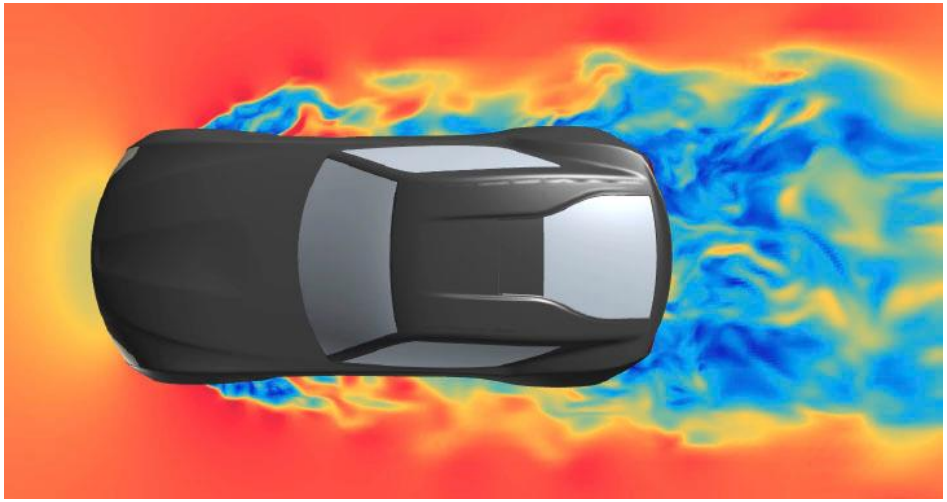
Simulation

From Wikipedia, the free encyclopedia

Not to be confused with Stimulation.

"Simulator" redirects here. For other uses, see Simulator (disambiguation).

Simulation is the imitation of the operation of a real-world process or system over time.[1] The act of simulating something first requires that a model be developed; this model represents the key characteristics, behaviors and functions of the selected physical or abstract system or process. The model represents the system itself, whereas the simulation represents the operation of the system over time.



Simulators

Types of simulators

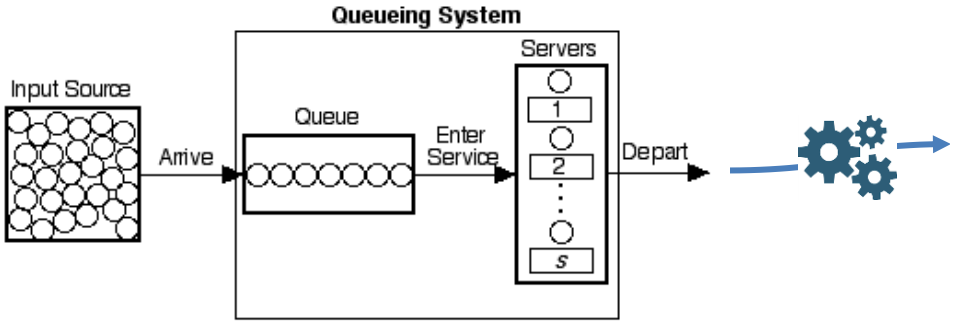
Static	vs	**Dynamic**
Continuous	vs	**Discrete**
Deterministic	vs	**Stochastic**

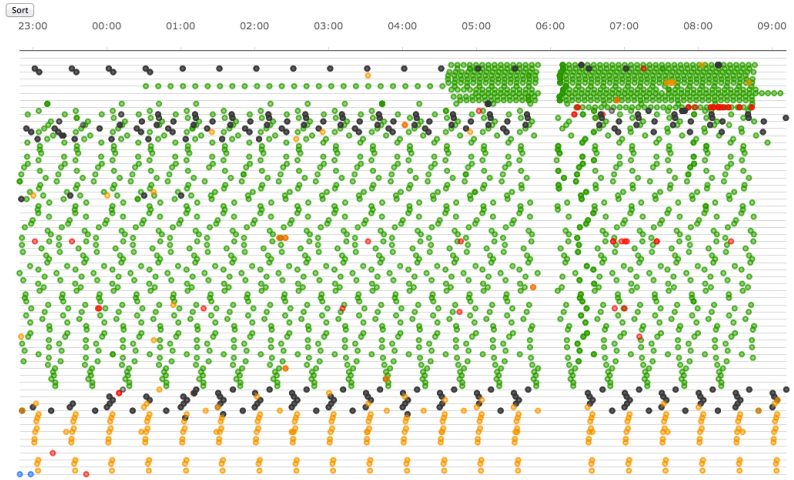
Simulators

So, if simulation is a game…

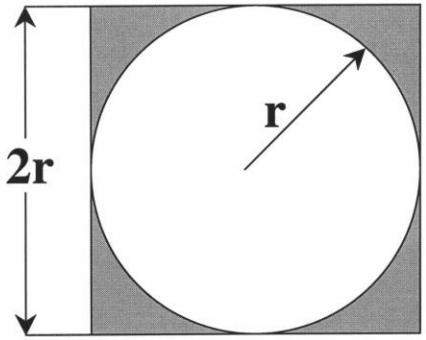
What are the rules ?

From a model to a game playing:





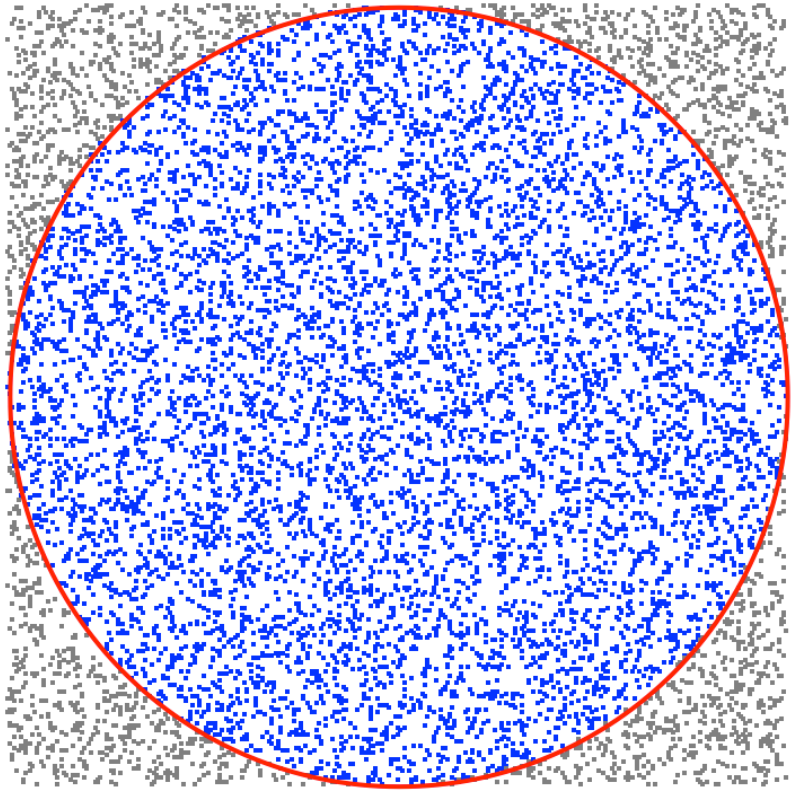
Static simulations – the rules



Area of Square = $4r^2$
Area of Circle = πr^2
Ratio of area of Circle to area of Square = $\pi r^2/4r^2$
= $\pi/4$

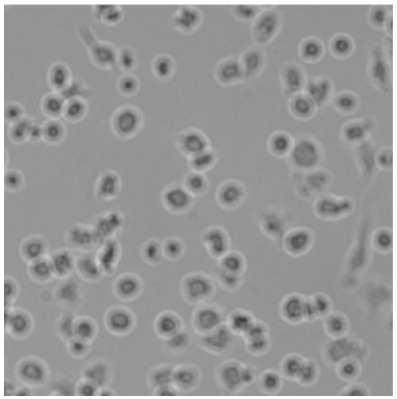
Total number of throws = N
No. hits inside circle = M
Ratio of no. hits inside circle to total no. throws = M/N

$\pi/4 \approx M/N \Rightarrow \pi \approx 4 * M/N$

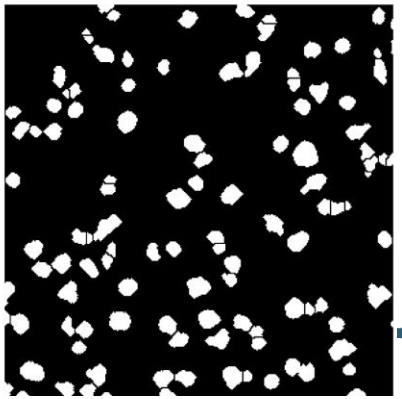


Static simulations – Playing the game

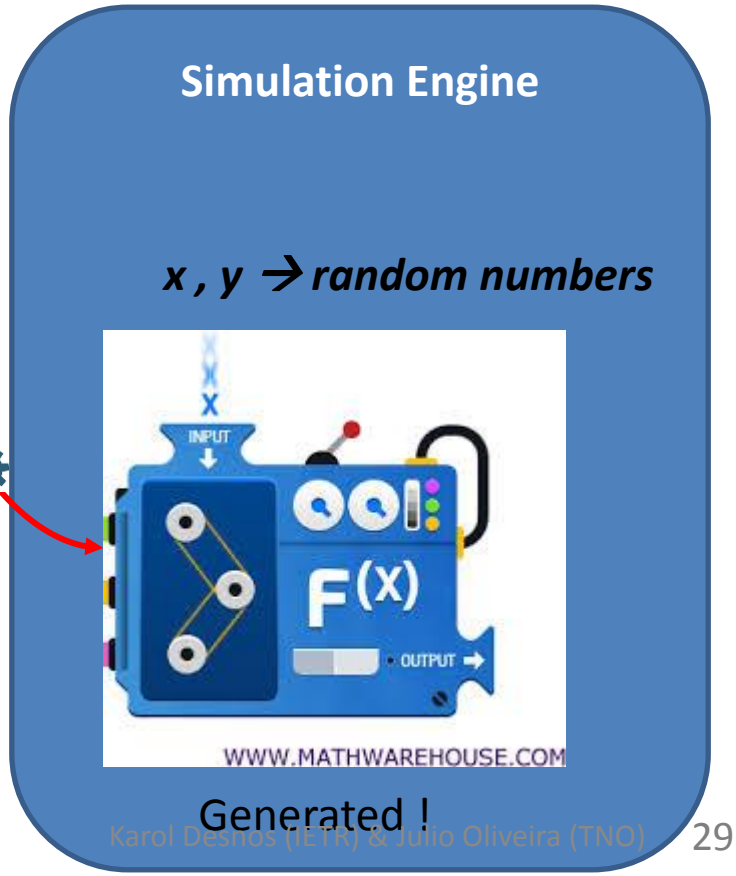
Can you play the static simulation game to calculate the density of blood cells in a sample?



Reality



Model

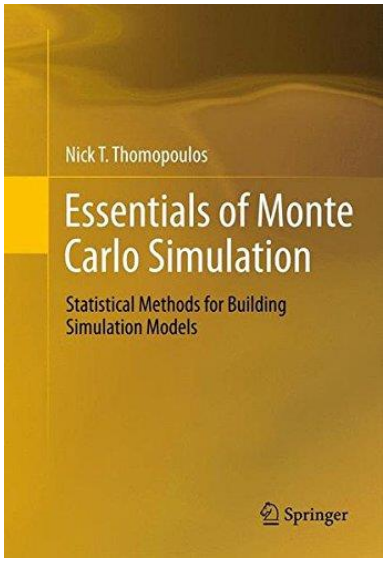


Simulators

Static simulators

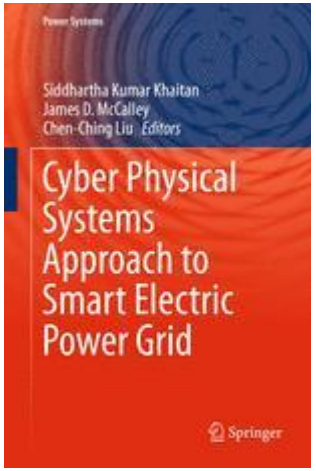
GoldSim

https://www.goldsim.com

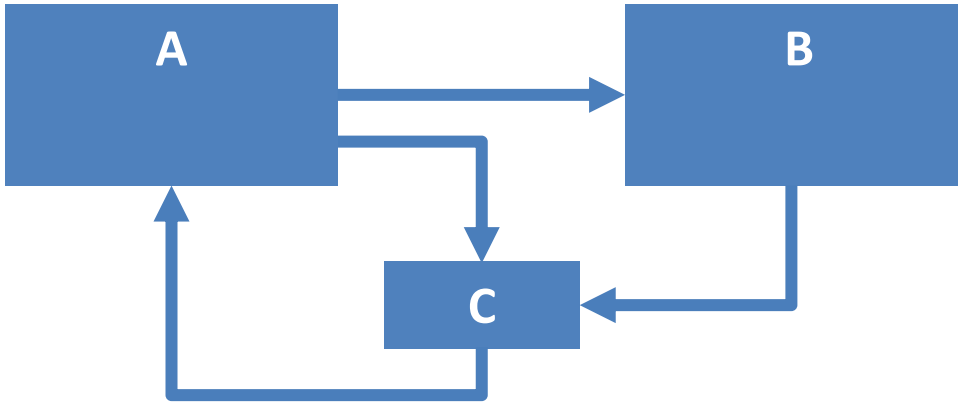


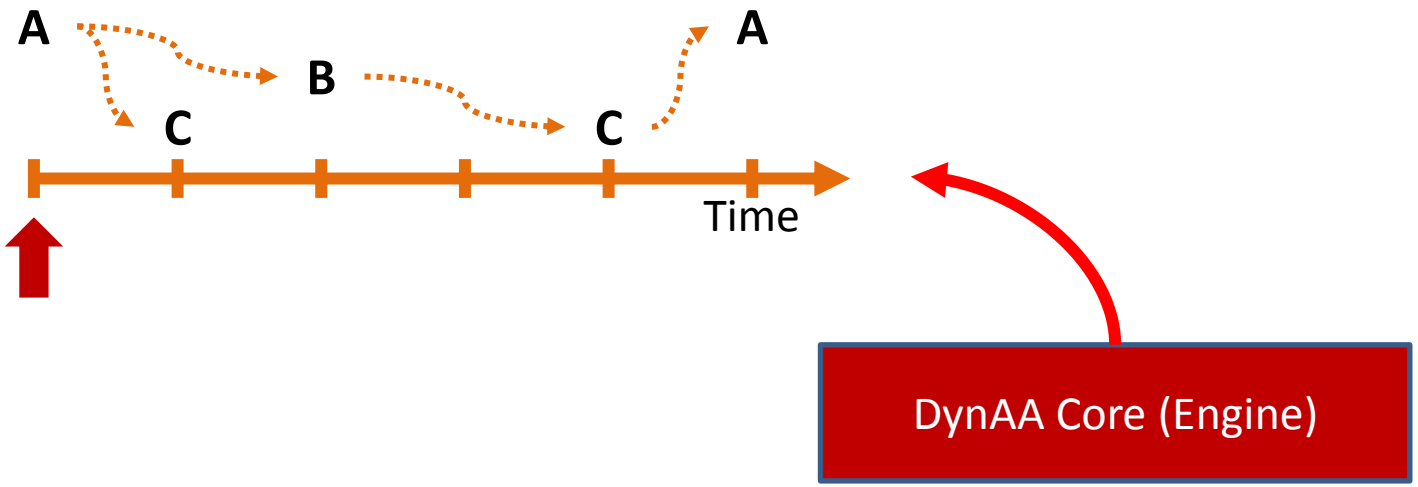
MATLAB

https://www.mathworks.com



Discrete event simulation – the rules

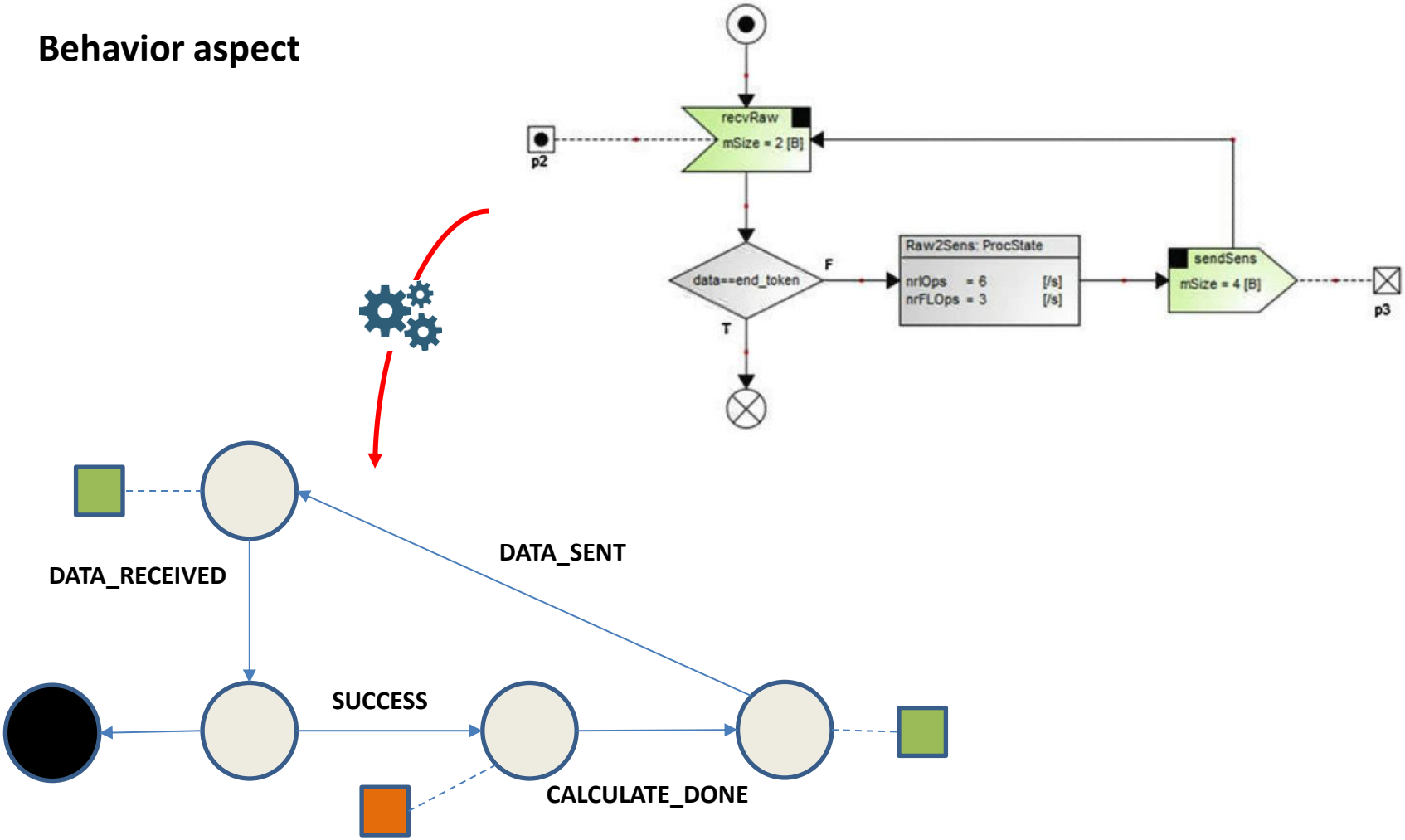




Source : Karol Desnos

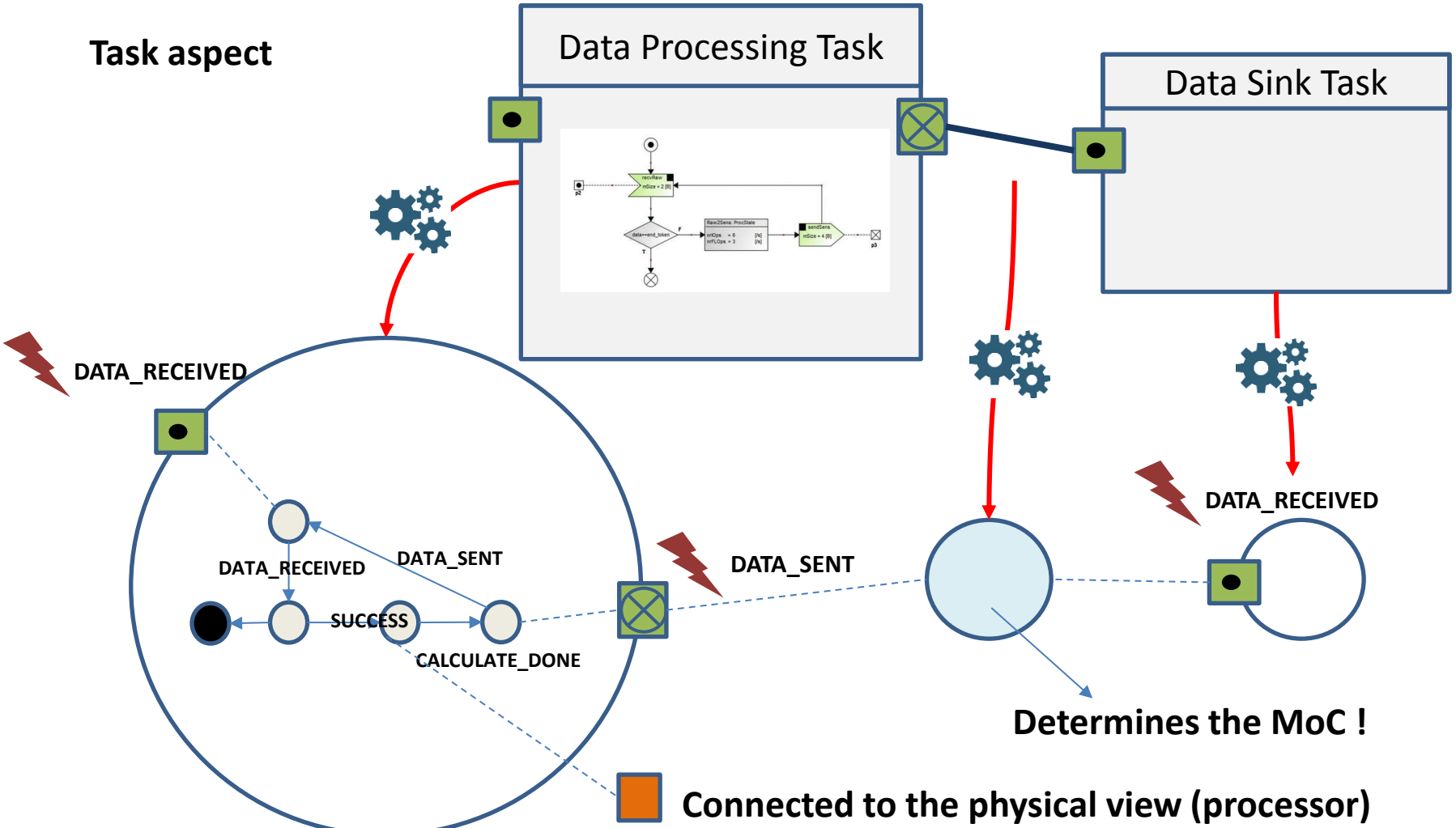
DES – Playing the game : An example from DynAA

Behavior aspect



Simulators

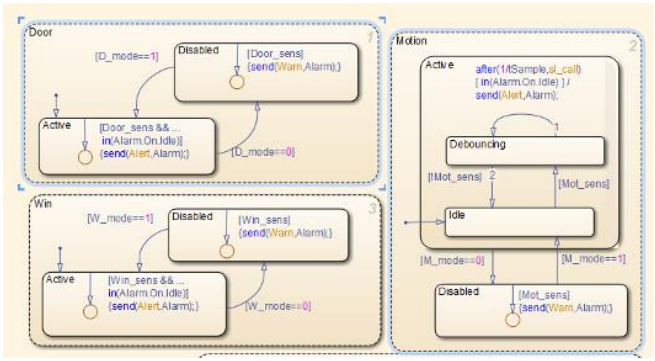
DES – Playing the game : An example from DynAA



Simulators

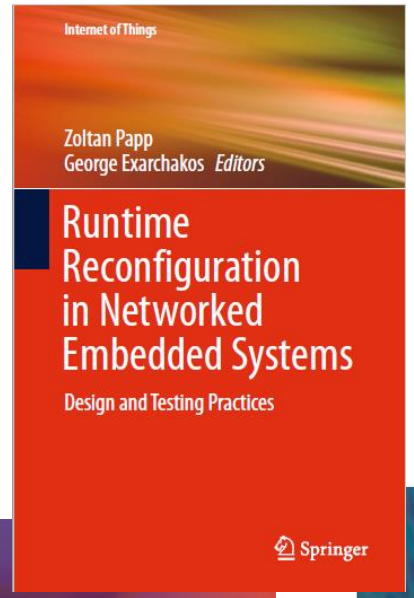
Discrete event simulators

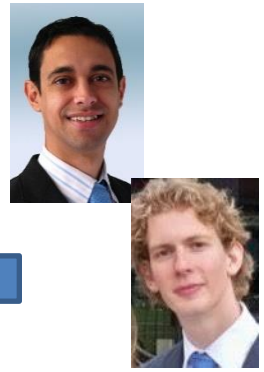
DynAA !!!!



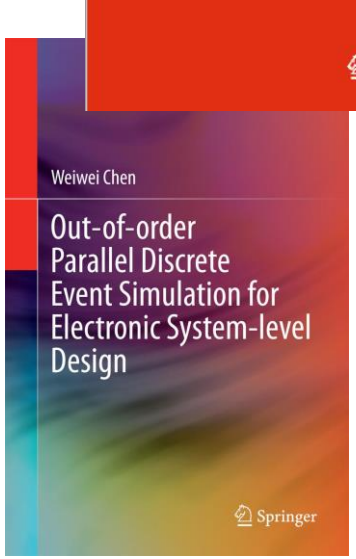
MATLAB

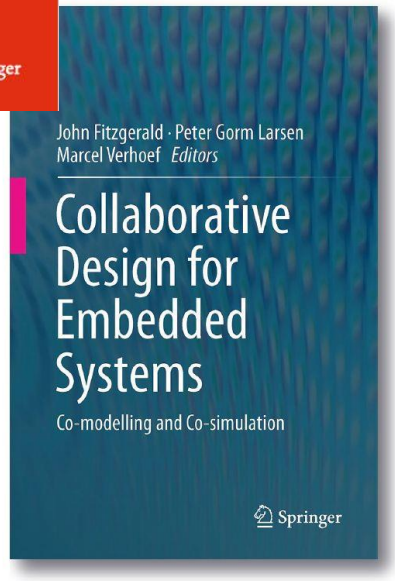
https://www.mathworks.com



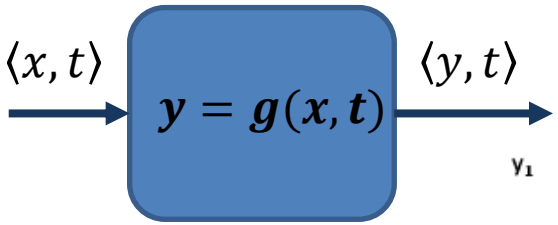


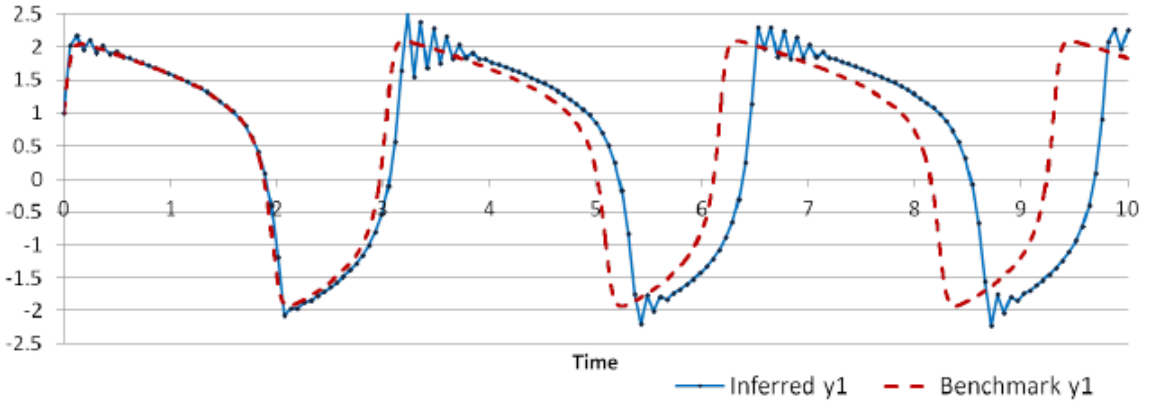
Find us Inside !



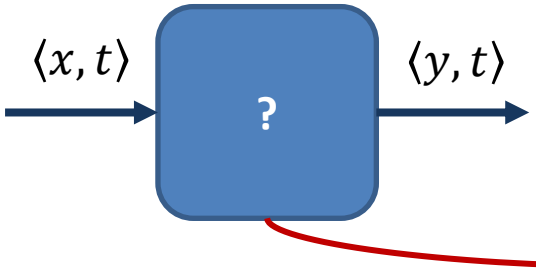


Continuous simulations – the rules



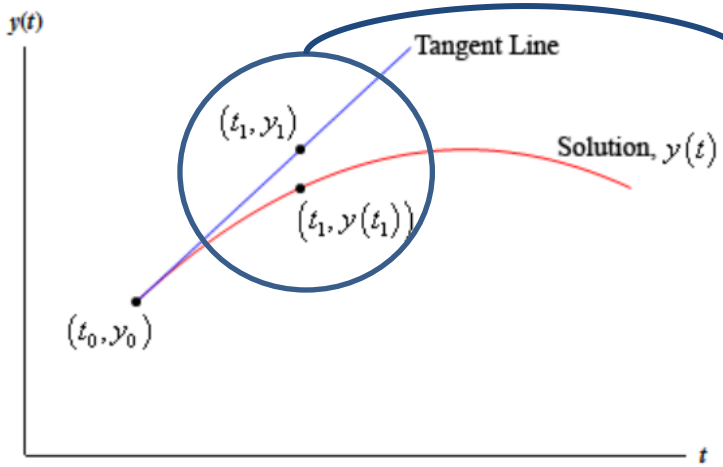


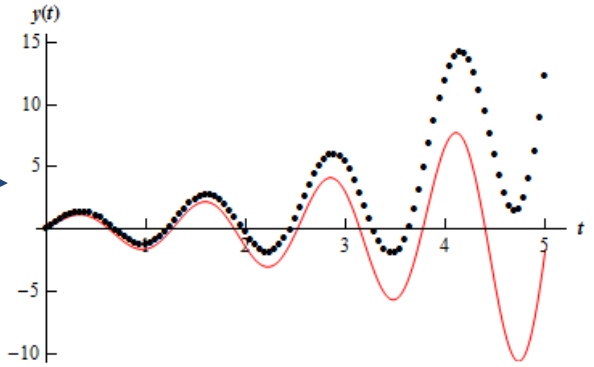
Play the game by advancing the time, step by step, and evaluating the function



$$f(t,y) = \frac{dy}{dt}$$

Continuous simulations – the rules

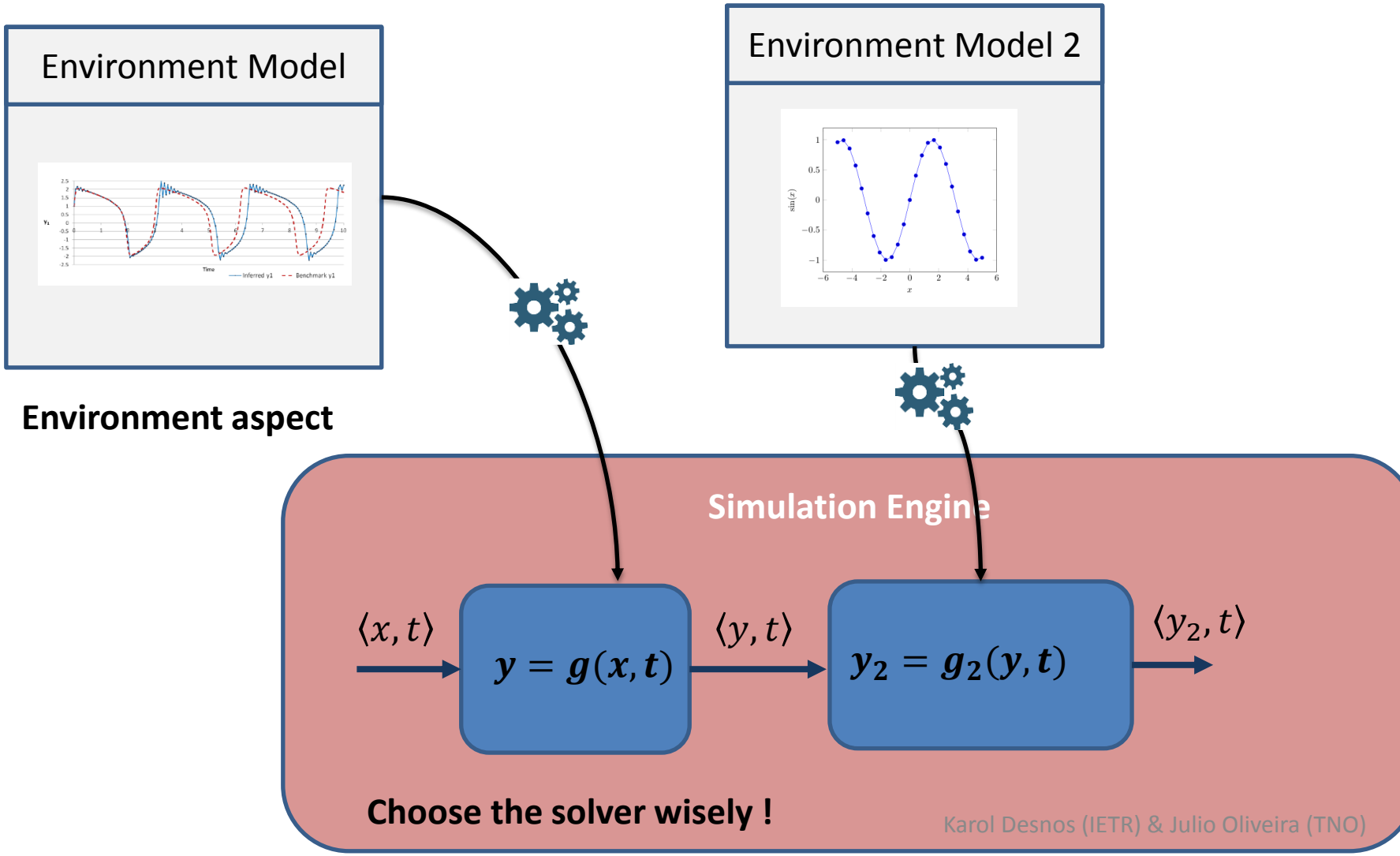




Euler's method : the simplest solver

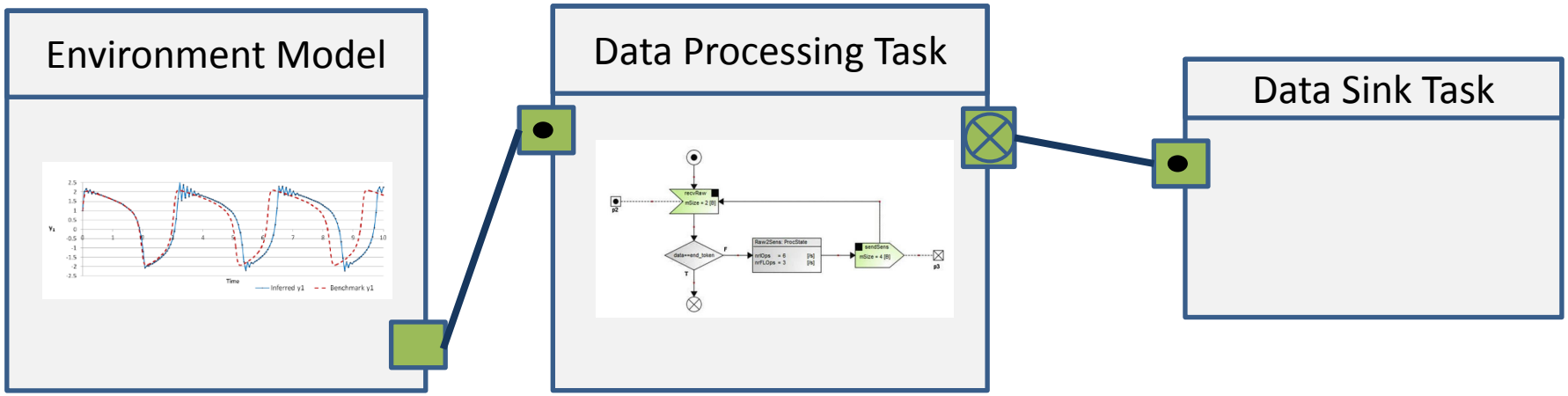
1. **define** $f(t, y)$.
2. **input** t_0 and y_0.
3. **input** step size, h and the number of steps, n.
4. **for** j from 1 to n **do**
   a. $m = f(t_0, y_0)$
   b. $y_1 = y_0 + h*m$
   c. $t_1 = t_0 + h$
   d. Print t_1 and y_1
   e. $t_0 = t_1$
   f. $y_0 = y_1$
5. **end**

Continuous simulation – Playing the game

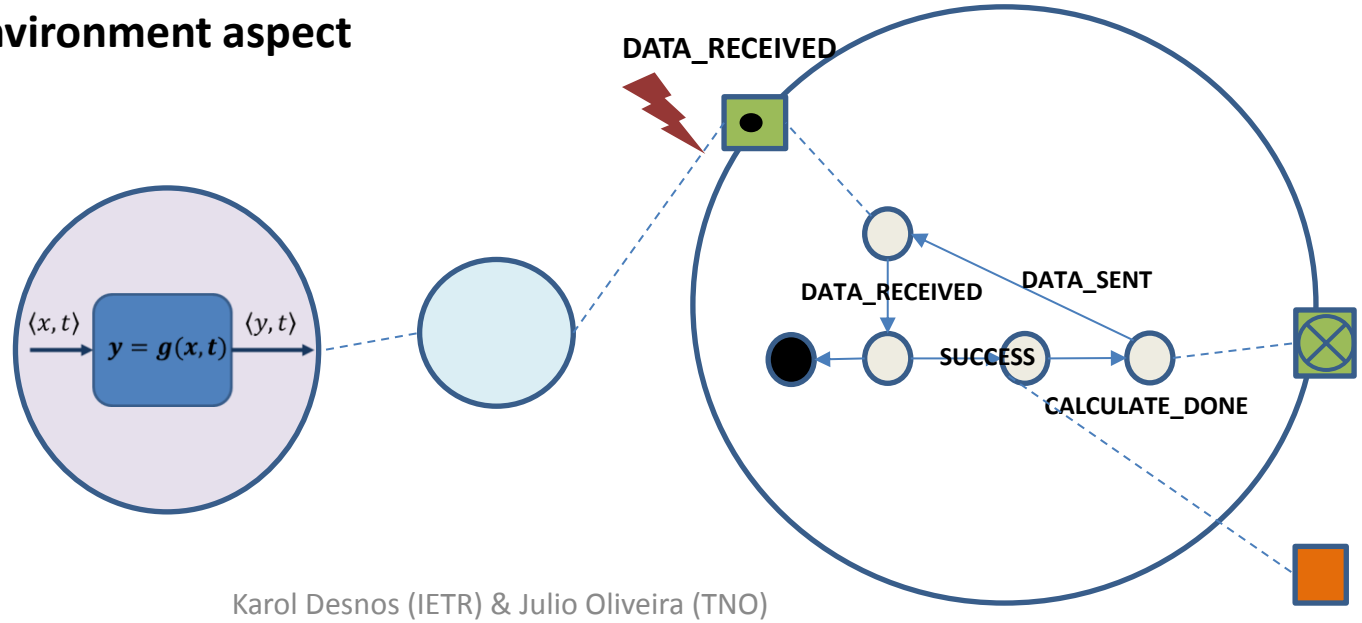


Simulators

Short view on Hybrid simulation



Environment aspect

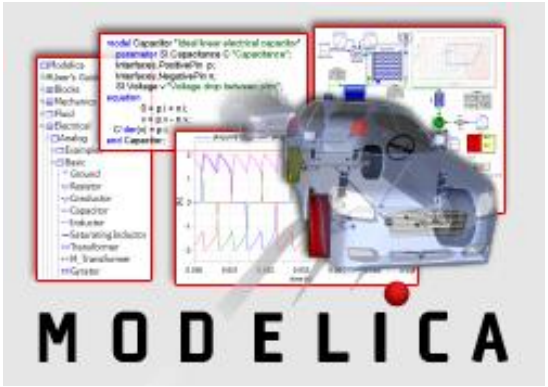


Looks the same, feels the same, but much more difficult !!!

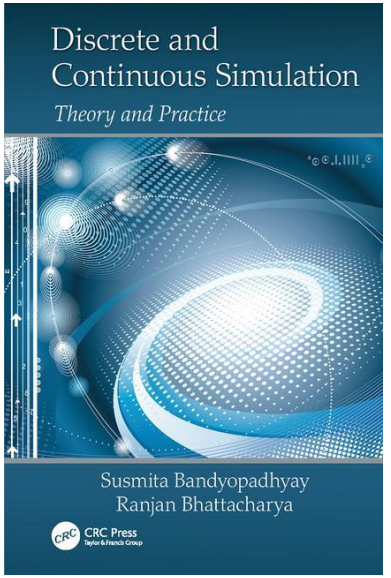
Video !

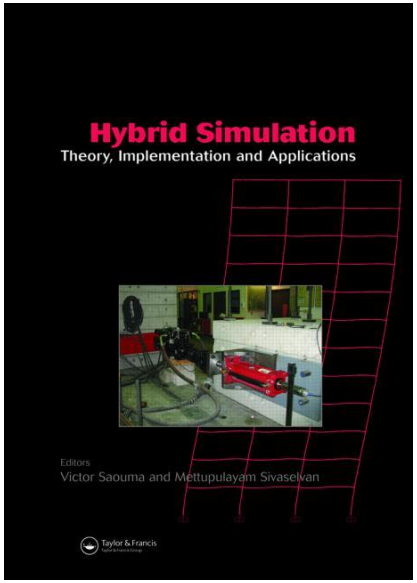
Simulators

Continuous (Hybrid) simulators



https://www.modelica.org/





MATLAB®
SIMULINK®

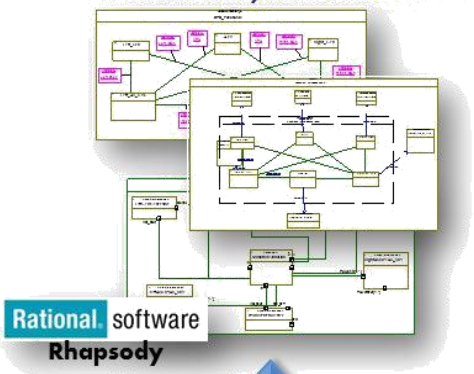
https://www.mathworks.com

HW/SW Cyber-System Modeling Tools

1. Modeling Environment and Languages
2. (HW &) SW Synthesis
3. Simulators
4. Analysis / Optimization
5. Runtime Support

Analysis / Optimization

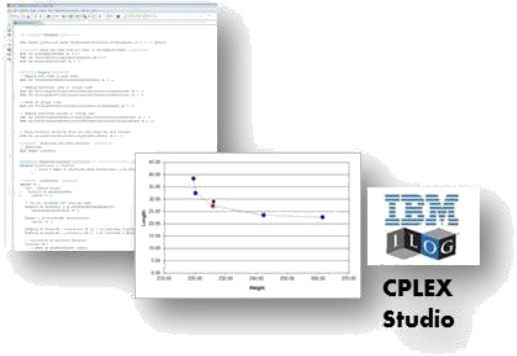
1. Describe system through different SysML views, including design alternatives, constraints and goals



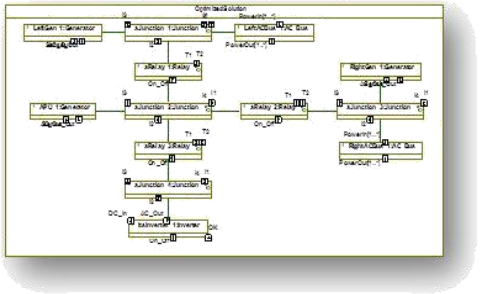
2. Derived Data Schema for Input and output structures

Architecture Optimization through Design Space Exploration iterations

3. Automatic translation(via an interchange format) into Optimization solver



4. Optimized architecture back annotated to SysML model



Analysis / Optimization

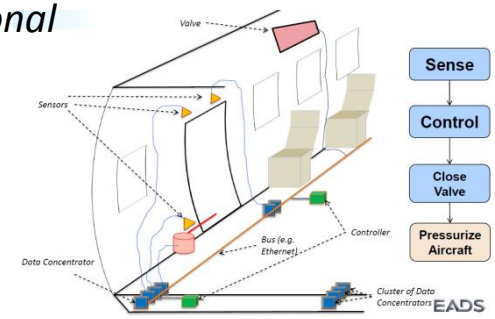
Development and Analysis of a Simplified Doors Control System

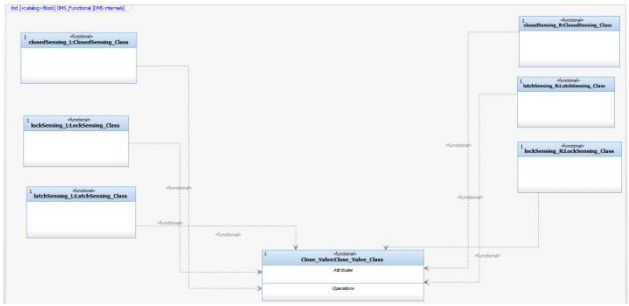
- Monitor and Control Passenger Doors, Emergency Exits, and Cargo Doors
- Design a system out of existing components for best weight, cost, power etc.

Analysis / Optimization

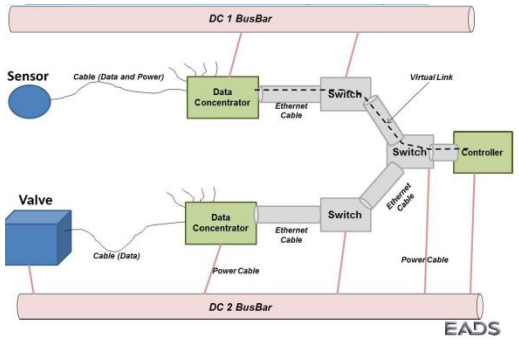
From user story to model

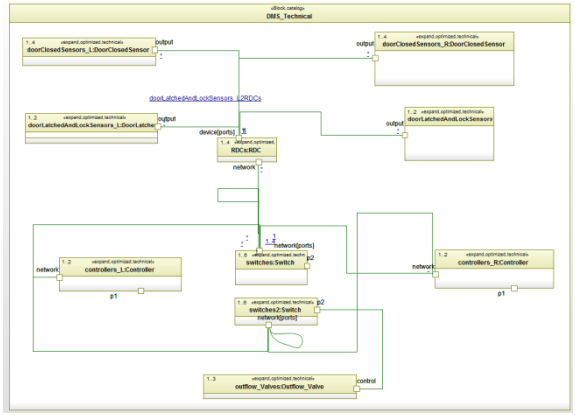
Functional



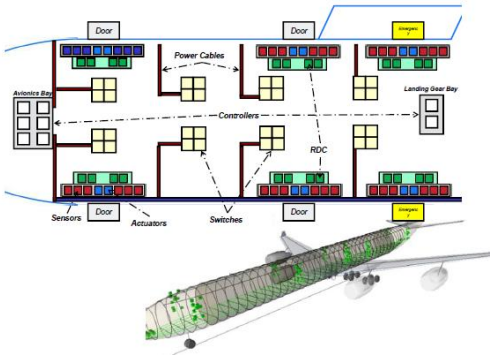


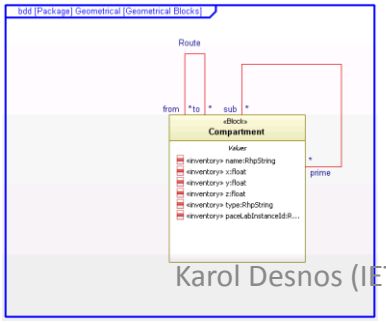
Technical





Geometrical





Analysis / Optimization

Optimization Auto-Generation

P1: whenever [E] occurs [SC] holds during following [I]

P2: whenever [E1] occurs [E2] implies [E3] during following [I]

P3: whenever [E1] occurs [E2] does not occur during following [I]

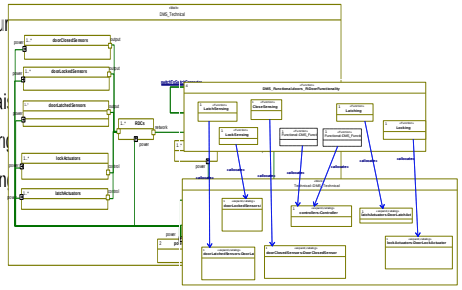
P4: whenever [E] occurs [E/SC] occur

P5: [SC] during [I] raises [E]

P6: [E1] occurs [N] times during [I] rai

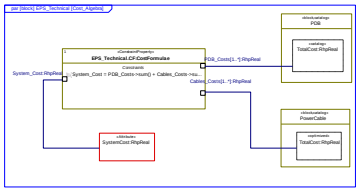
P7: [E] occurs at most [N] times durin

P8: [SC] during [I] implies [SC1] durin



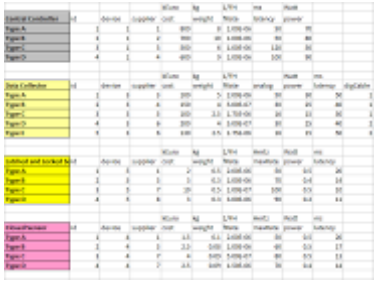
▪*Requirements & Constraints*

▪*Objectives*



▪Automatic Generation

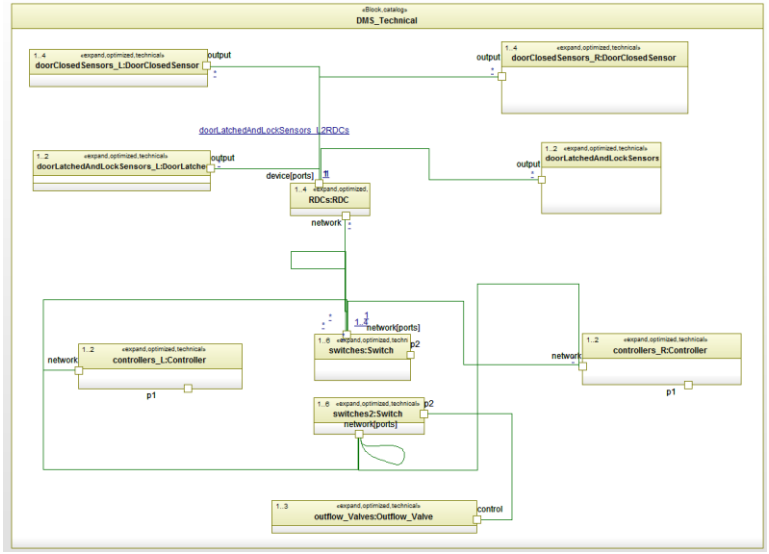
▪*Component library*



▪*Optimization Problem Formulation (CPLEX)*

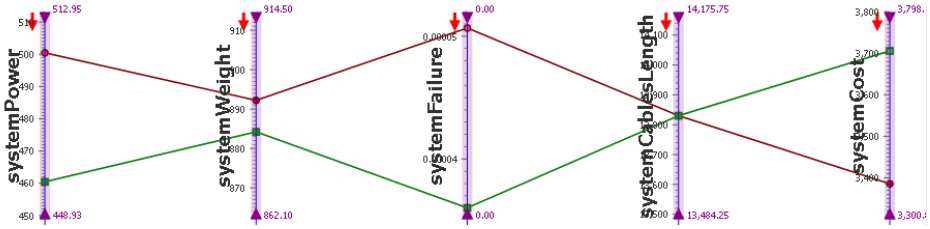
Analysis / Optimization

1. SysML modeling

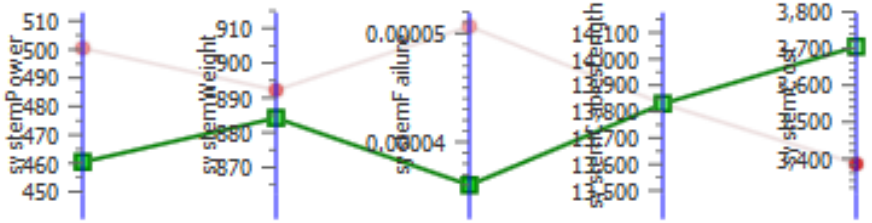


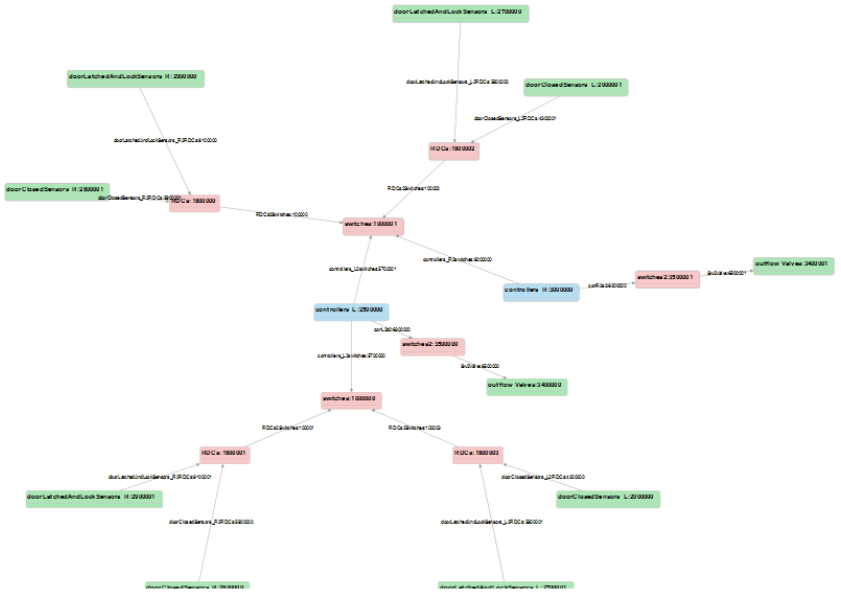
2. Run Optimization and show alternatives

ID	systemPower	systemWeight	systemFailure	systemCablesLength	systemCost	Finished at	Duration
2ece937c-a4b9-4ac8-b0ab-702de3a1c94a	500.44	892.20	5.066E-5	13830.00	3385.50	May 23, 2013 11:14:54 AM	942 sec
3232cbf2-0383-4500-811a-52d643079cd3	460.44	884.20	3.602E-5	13830.00	3705.50	May 23, 2013 11:10:54 AM	938 sec



Results





HW/SW Cyber-System Modeling Tools

1. Modeling Environment and Languages
2. (HW &) SW Synthesis
3. Simulators
4. Verification / Validation
5. Runtime Support

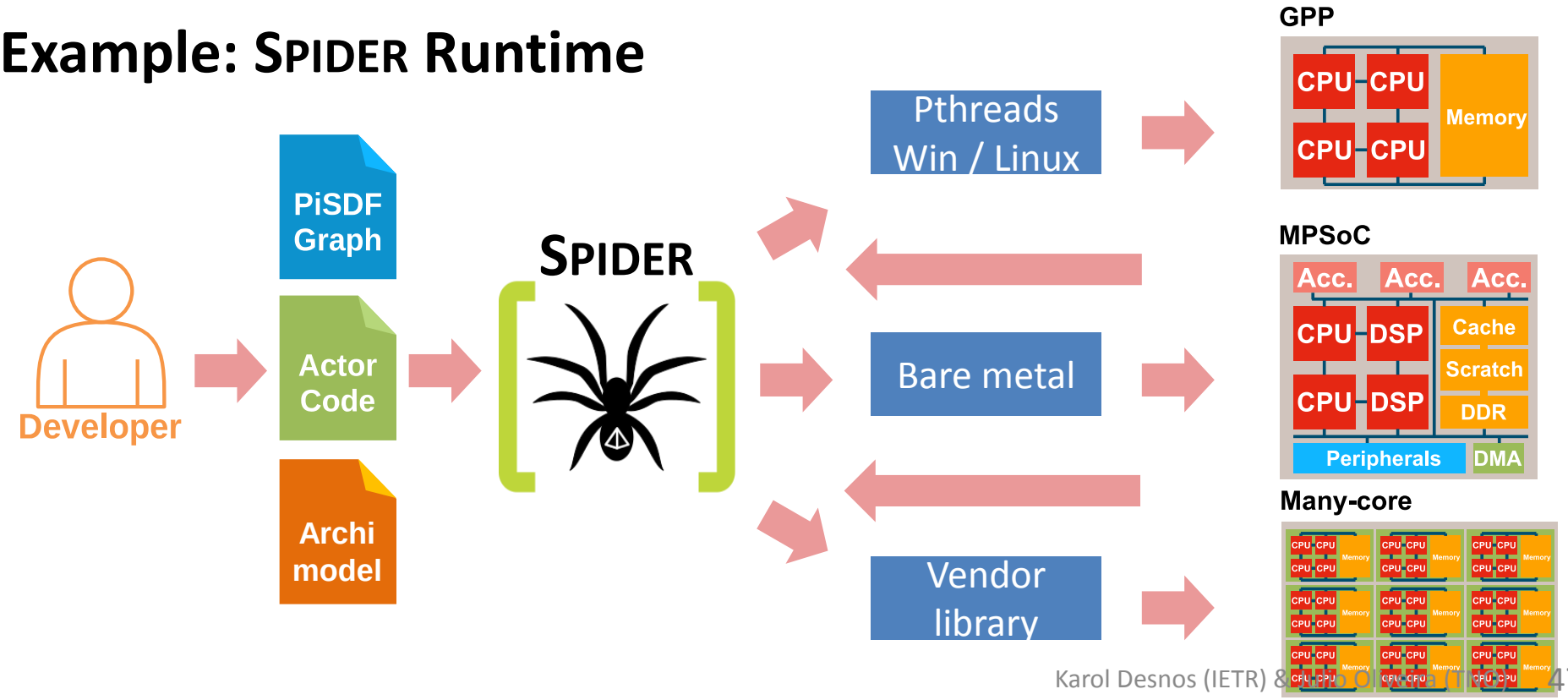
Runtime support

Overview

Runtime adaptation layer:

- Services needed to run an application
- Provided by “classic” OS, and/or model-aware utilities.

Example: SPIDER Runtime

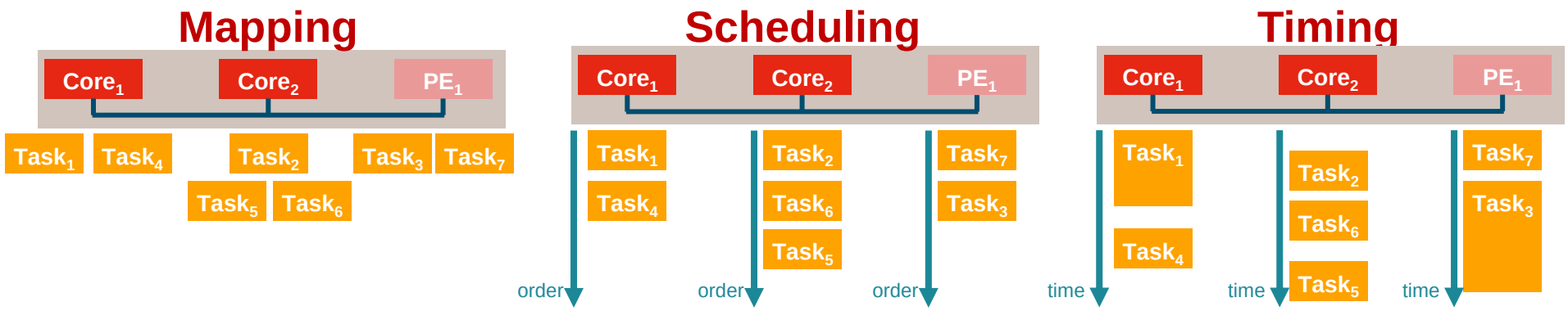


Different Runtime Strategies

Adaptivity++

	Mapping	Scheduling	Timing
fully dynamic	run	run	run
static-assignment	compile	run	run
self-timed	compile	compile	run
fully static	compile	compile	compile

Performance++



Source: E. Lee, "Scheduling Strategies for Multiprocessor real-time DSP"

Runtime support

Fully static runtime support

Computation

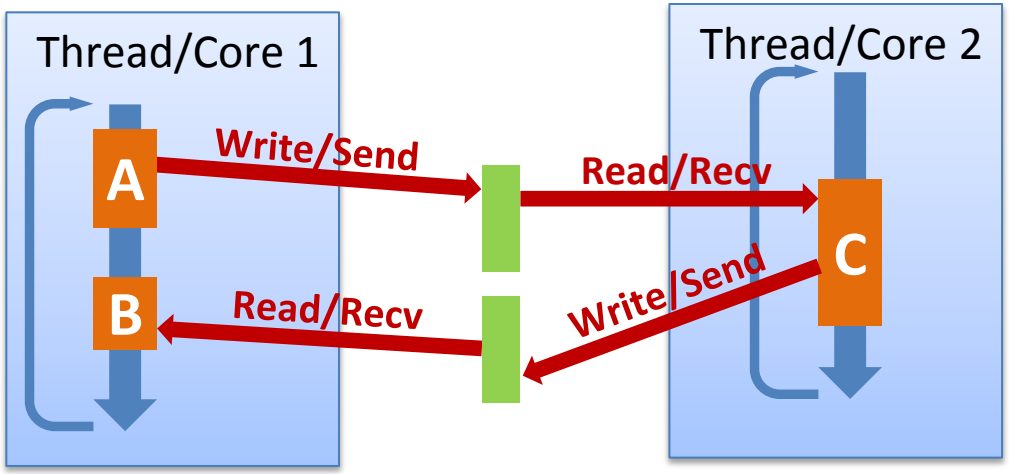
- Non-preemptible processes/threads
- Bare-metal or real-time OS
- Global clock

Communications

- Non-blocking

Used for:

- Hard Real-time systems



Runtime support

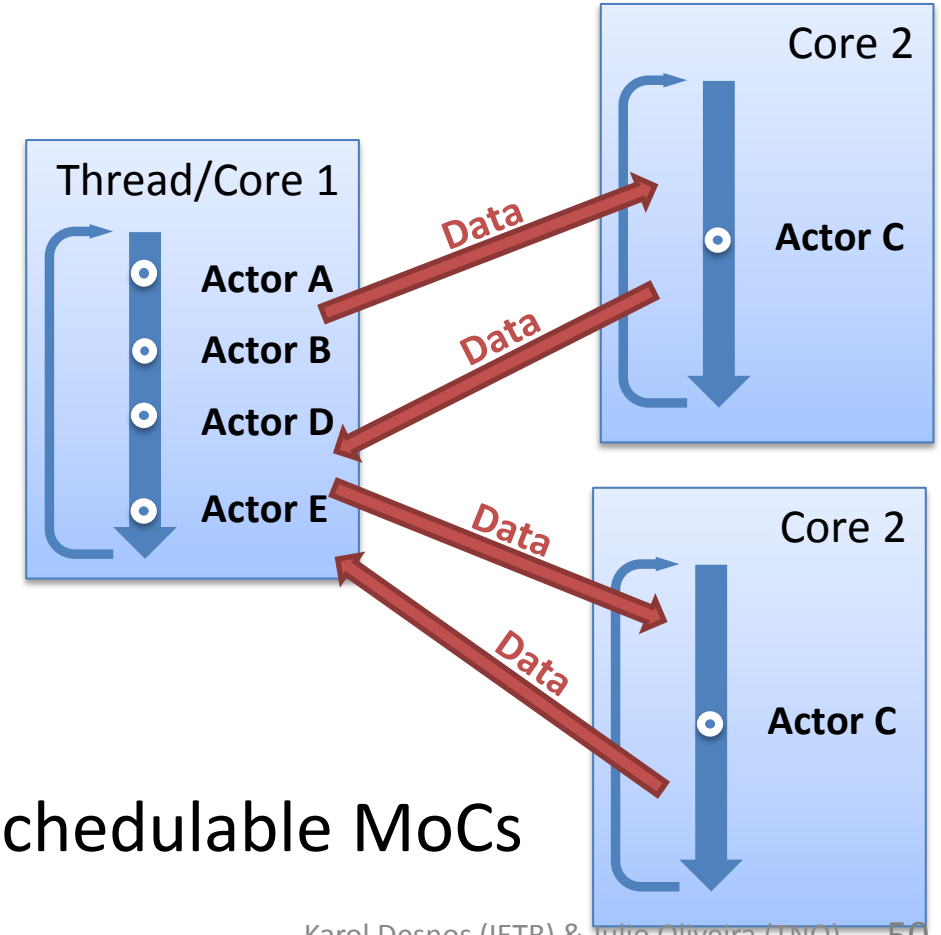
Self-timed runtime support

Computations

- Processes/threads/tasks
- Bare-metal or OS

Communications

- Blocking
- Synchronization mechanism:
  Barrier, Mutexes,
  Semaphores, …



Used for/by:

- Real-time systems, statically schedulable MoCs
- E.g. Preesm, Syndex

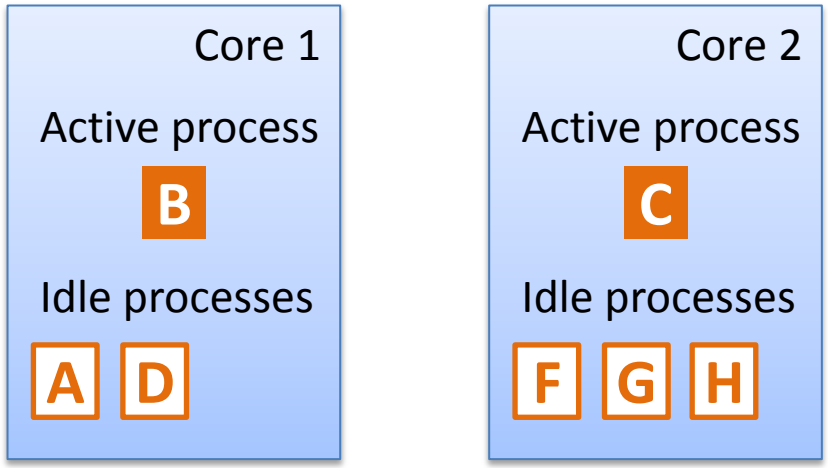
Static assignment runtime support

Computations

- Processes/threads/tasks
- OS with scheduler

Communications

- Blocking
- Synchronization mechanism:
  Barrier, Mutexes,
  Semaphores, Yield…



Used for/by:

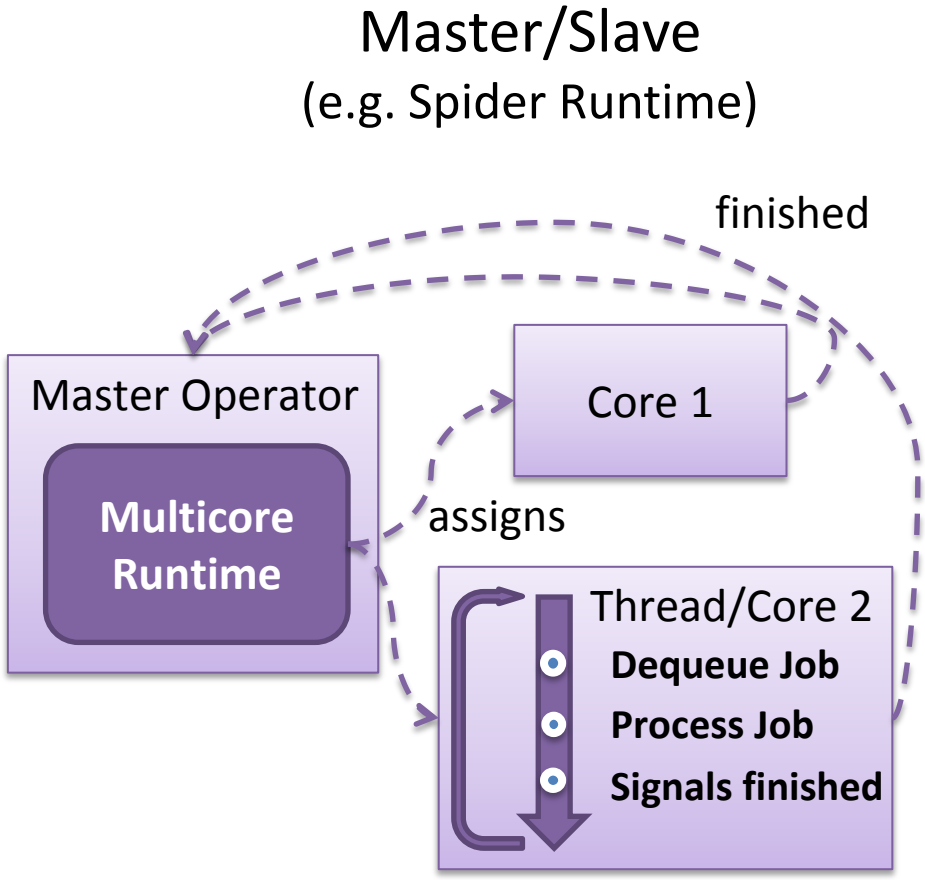
- Dynamic/Reconfigurable models (e.g. Kahn Process Network)
- E.g. Open RVC-CAL Compiler (ORCC), PaDaF

Runtime support

Fully dynamic runtime supports

Master/Slave model

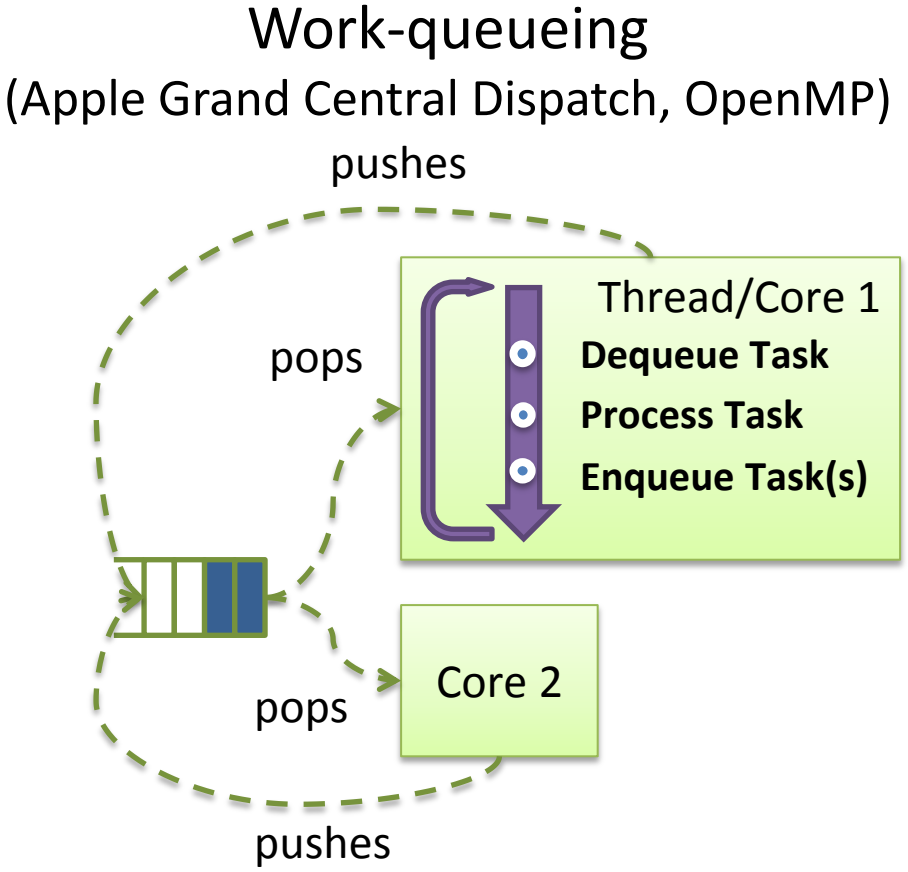
- Pros:
  - Adaptivity
  - Controlled decisions
  - Compatible with heterogeneous target
- Cons:
  - Master overhead
  - Master bottleneck



Fully dynamic runtime supports

Work queuing

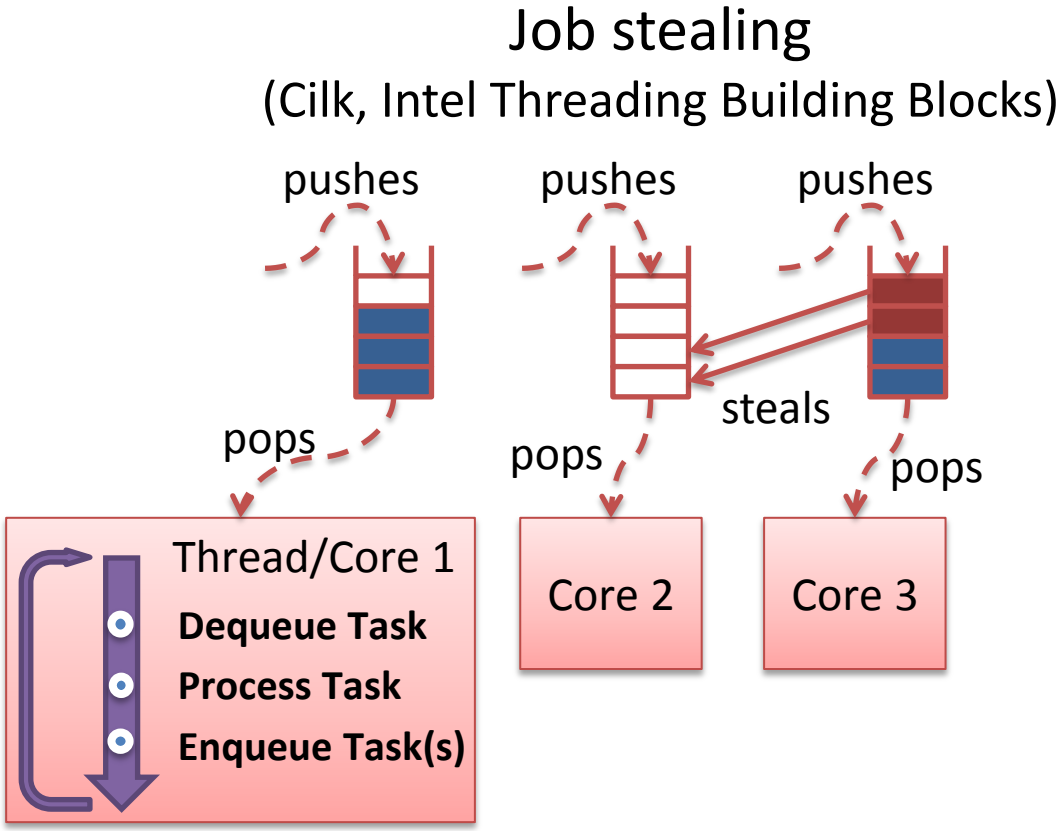
- Pros:
  - Adaptivity
  - Light/No Overhead
- Cons:
  - Shared queue bottleneck
  - No “intelligence”
  - Non-determinist



Fully dynamic runtime supports

Job Stealing

- Pros:
  - Adaptivity
  - Light/No Overhead
  - No bottleneck
- Cons:
  - No “intelligence”
  - Non-determinist



Final words


AN x64 PROCESSOR IS SCREAMING ALONG AT BILLIONS OF CYCLES PER SECOND TO RUN THE XNU KERNEL, WHICH IS FRANTICALLY WORKING THROUGH ALL THE POSIX-SPECIFIED ABSTRACTION TO CREATE THE DARWIN SYSTEM UNDERLYING OS X, WHICH IN TURN IS STRAINING ITSELF TO RUN FIREFOX AND ITS GECKO RENDERER, WHICH CREATES A FLASH OBJECT WHICH RENDERS DOZENS OF VIDEO FRAMES EVERY SECOND
BECAUSE I WANTED TO SEE A CAT JUMP INTO A BOX AND FALL OVER.
I AM A GOD.