

Using Concurrency Modeling Formalisms within System-on-a-Programmable-Chip Design

**Luís Gomes,
Anikó Costa,**

lugo@uninova.pt
akc@uninova.pt



Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia



LISBON - PORTUGAL

Introduction - I

General Framework:

Electrical and Computer Engineering Course

Final goal:

Co-design of embedded systems
using reconfigurable platforms
and SoC technologies

Embedded system =

Reactive system +

Real-time constraints +

Data processing capabilities

Introduction - II

The courses:

Integrated MSc course: 5-years long degree on
Electrical and Computer Engineering

(Regular) MSc course: 2-years long degree

The "Digital Systems Design" discipline :

Located at the 7th semester of Integrated MSc or
at the 1st semester of MSc course

Emphasis of Previous disciplines in Digital Systems:

Theory: Introductory digital systems; System
decomposition into control and data
parts; Microprocessor architectures.

Technology: From MSI ICs to CPLDs

Introduction - III

(Some) Courses on
Embedded Systems Education “Track”:

- Introductory Digital Systems Design
- Introductory Microprocessor Design **1st year**
 - 8-bit emphasis
 - 16/32 bit emphasis
- Real-time systems **3rd year**
- Digital System Design **4th year**
- Data Acquisition Systems **4th and 5th years**
- Advanced Courses

Some goals for “Digital Systems Design”

Theoretical:

Specification formalisms for concurrent systems:

Statecharts

Petri nets

Implementation:

Modularity and reusability of models

Usage of Hardware Description Languages (VHDL)

Technologies:

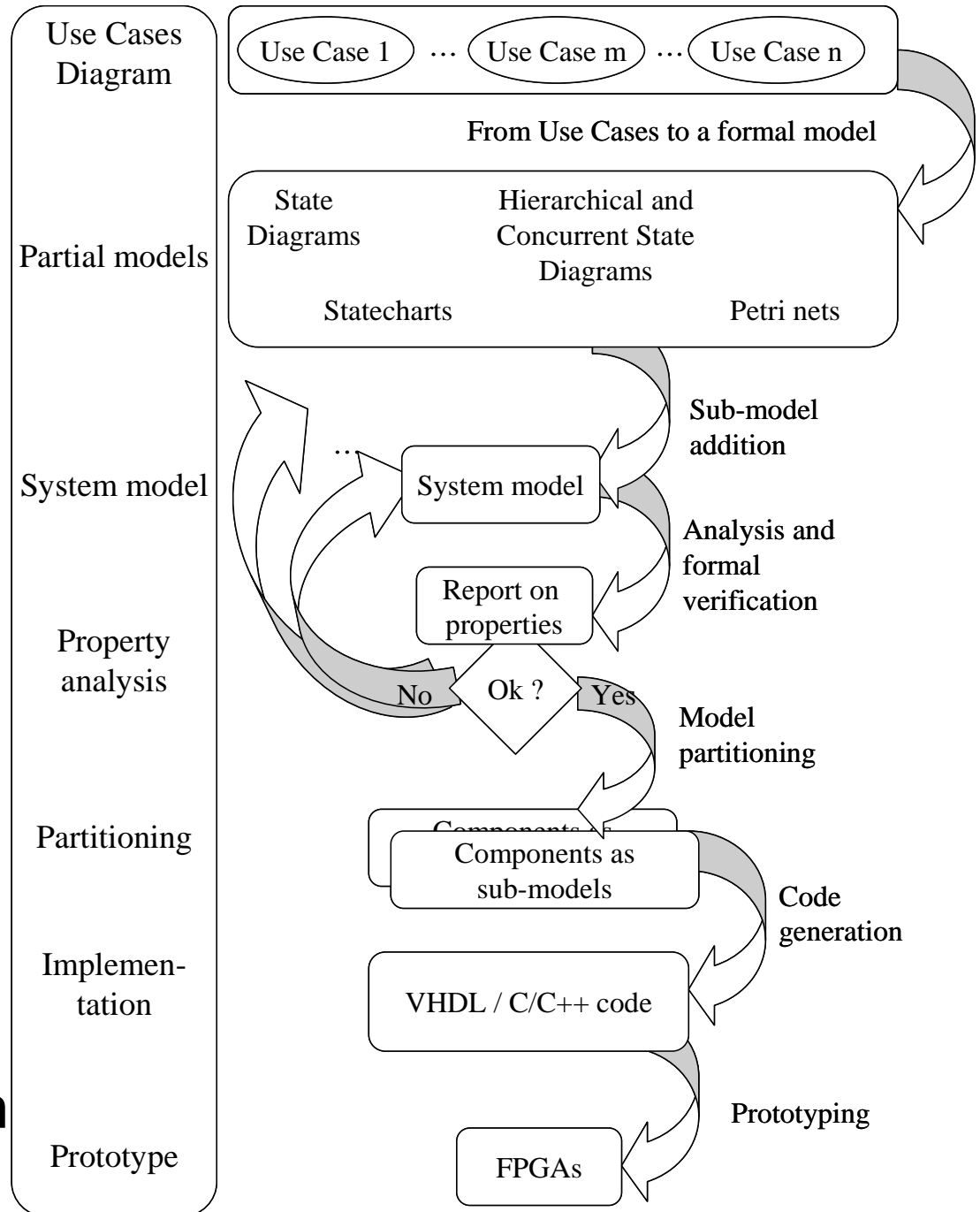
Usage of PLDs – Programmable Logic Devices
(CPLDs and FPGAs)

Structure of the presentation

- Introduction
- Proposed methodology overview:
 - hardware-software co-design
 - hardware-centric view
- Introducing concurrency formalisms through a running example
- Laboratory assignments plan
- Presentation of one "didactical" project
- Conclusions

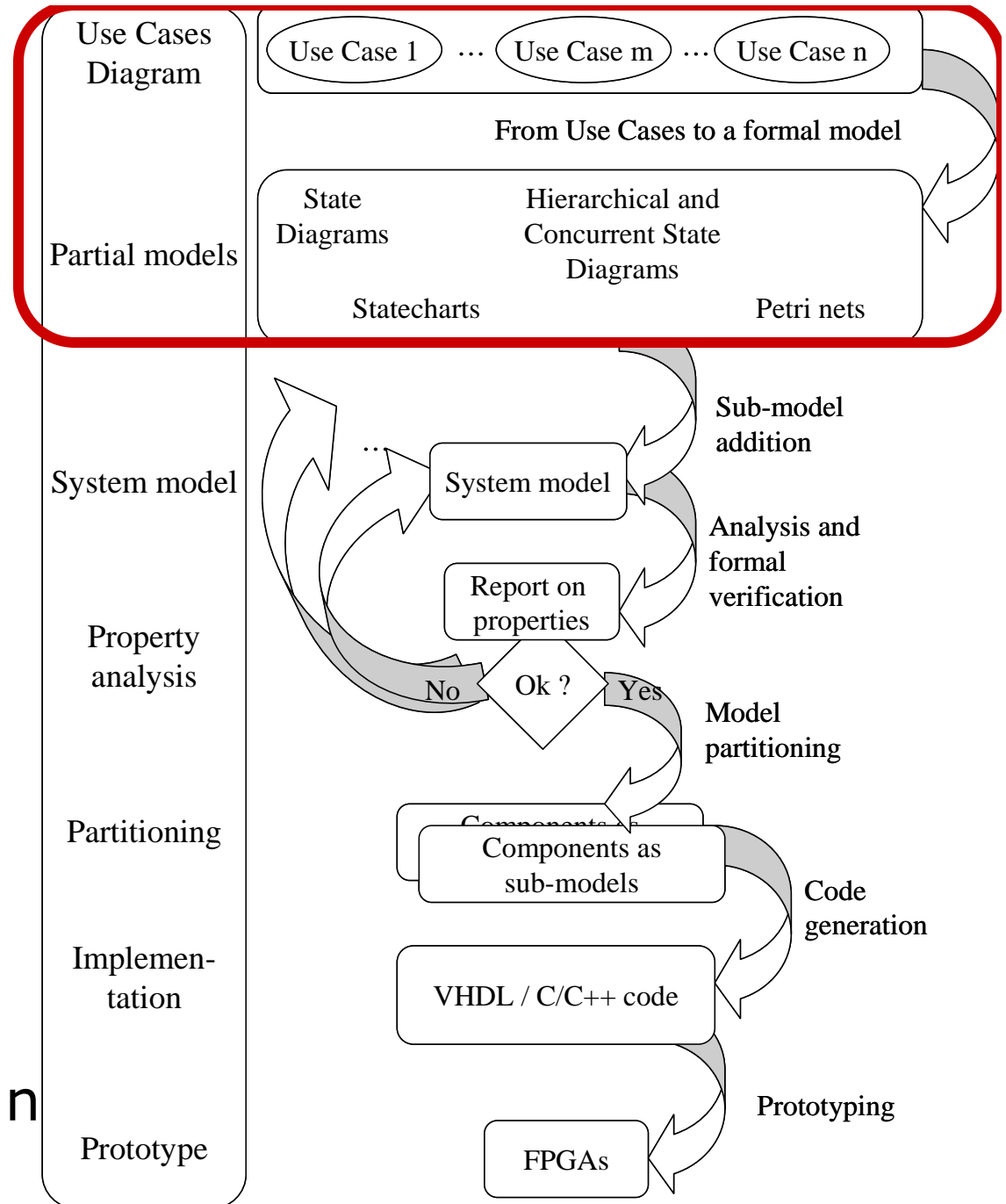
Methodology overview

"The hardware-
software co-design
view"



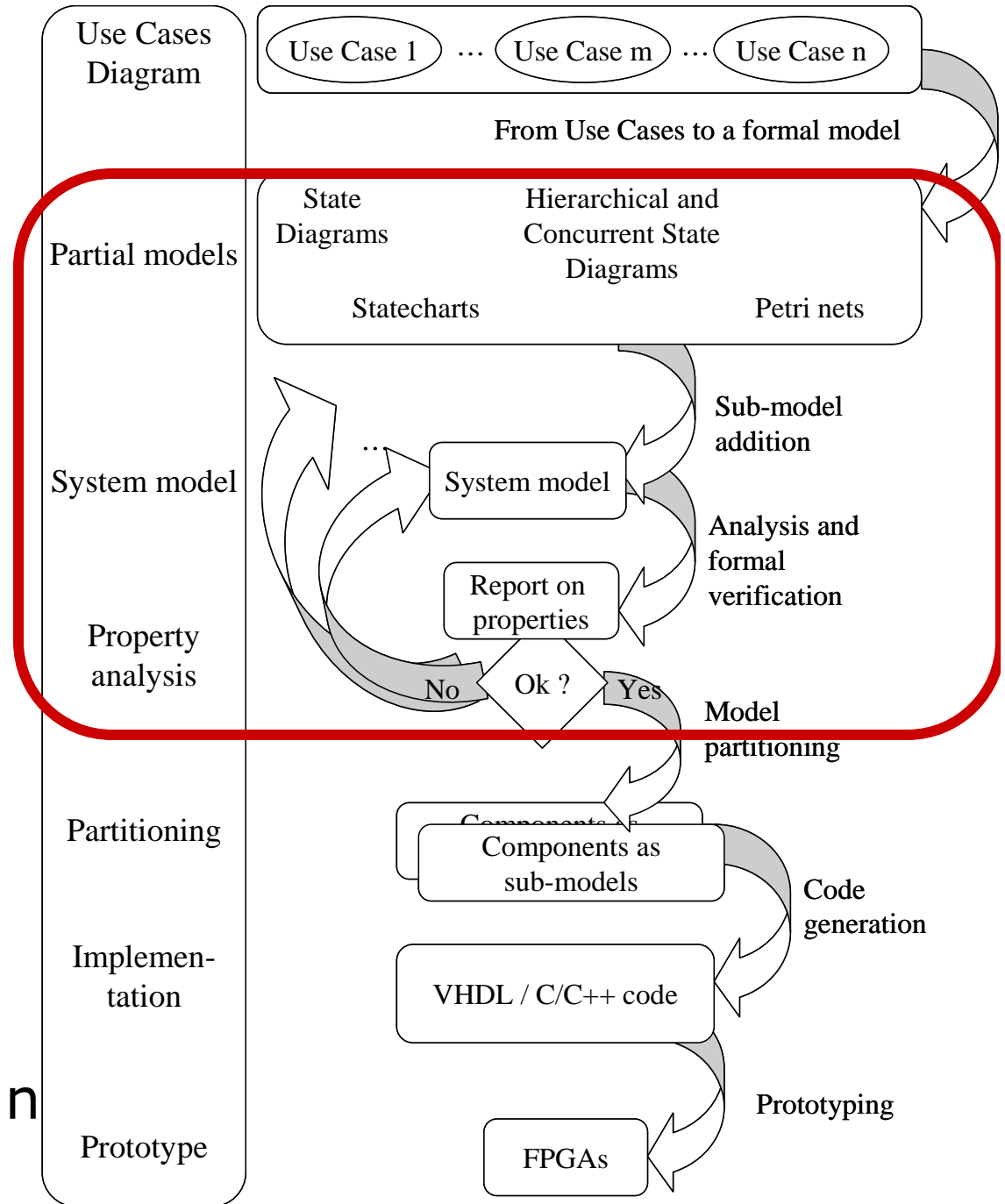
Methodology overview

"The hardware-
software co-design
view"



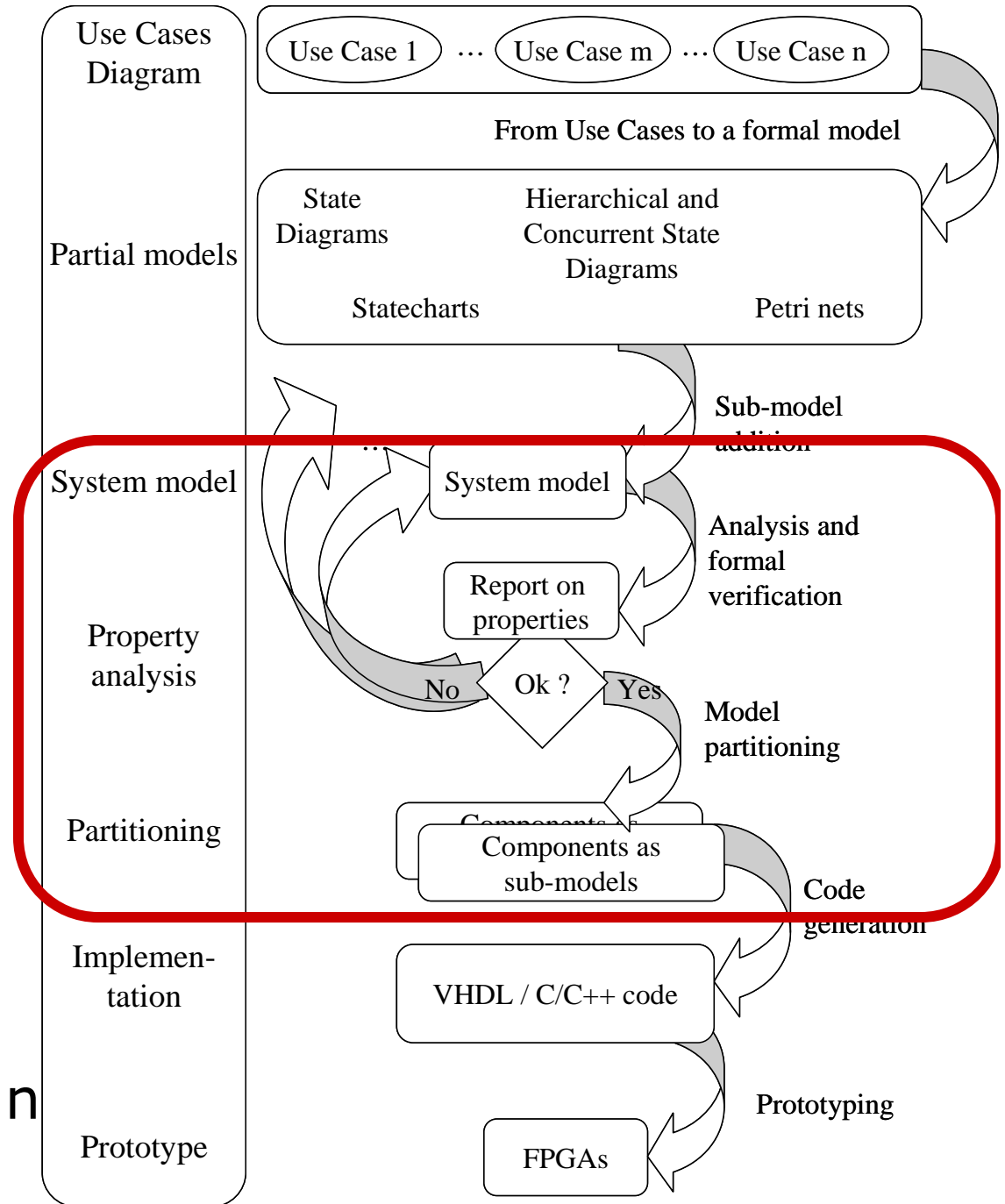
Methodology overview

"The hardware-
software co-design
view"



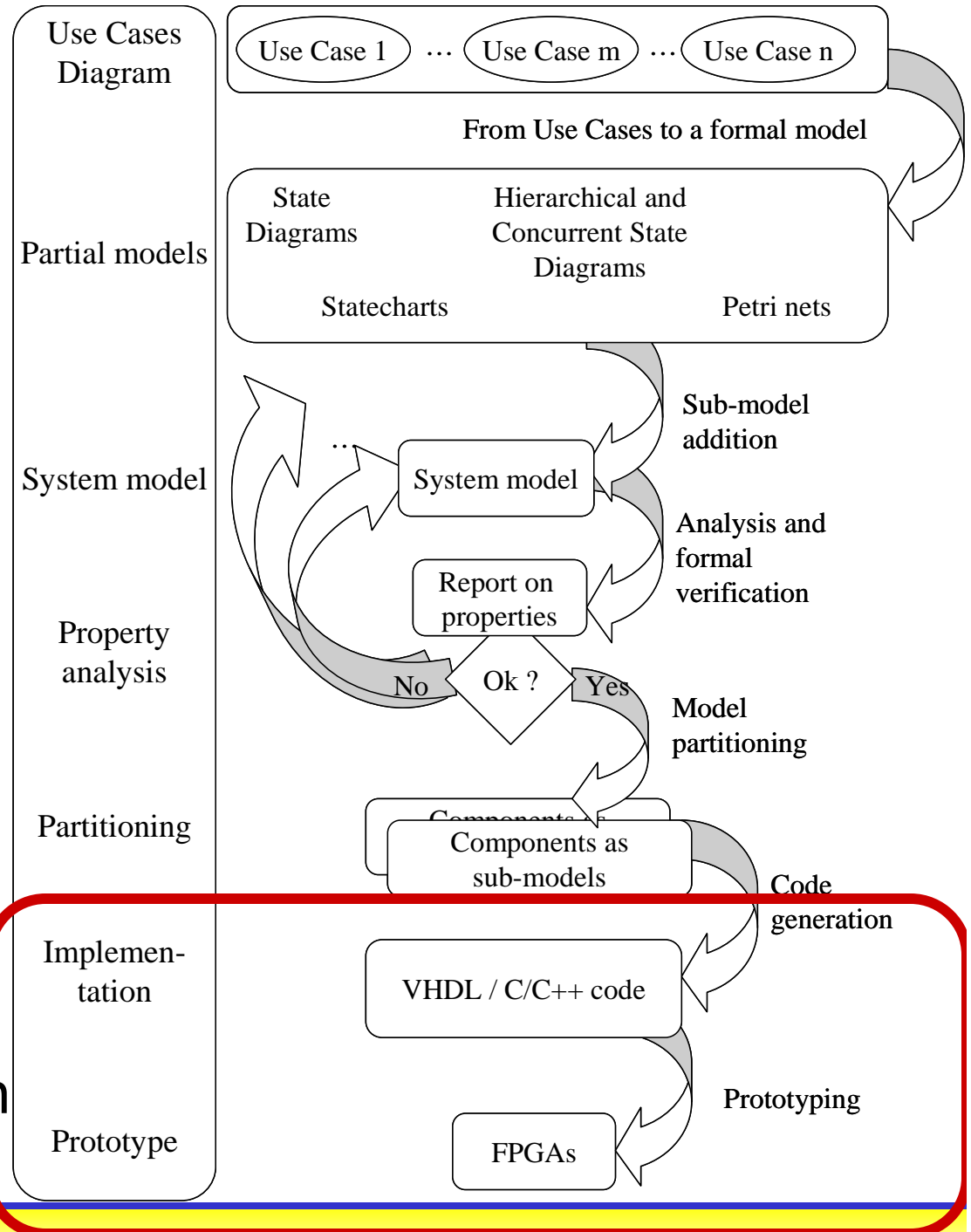
Methodology overview

"The hardware-
software co-design
view"



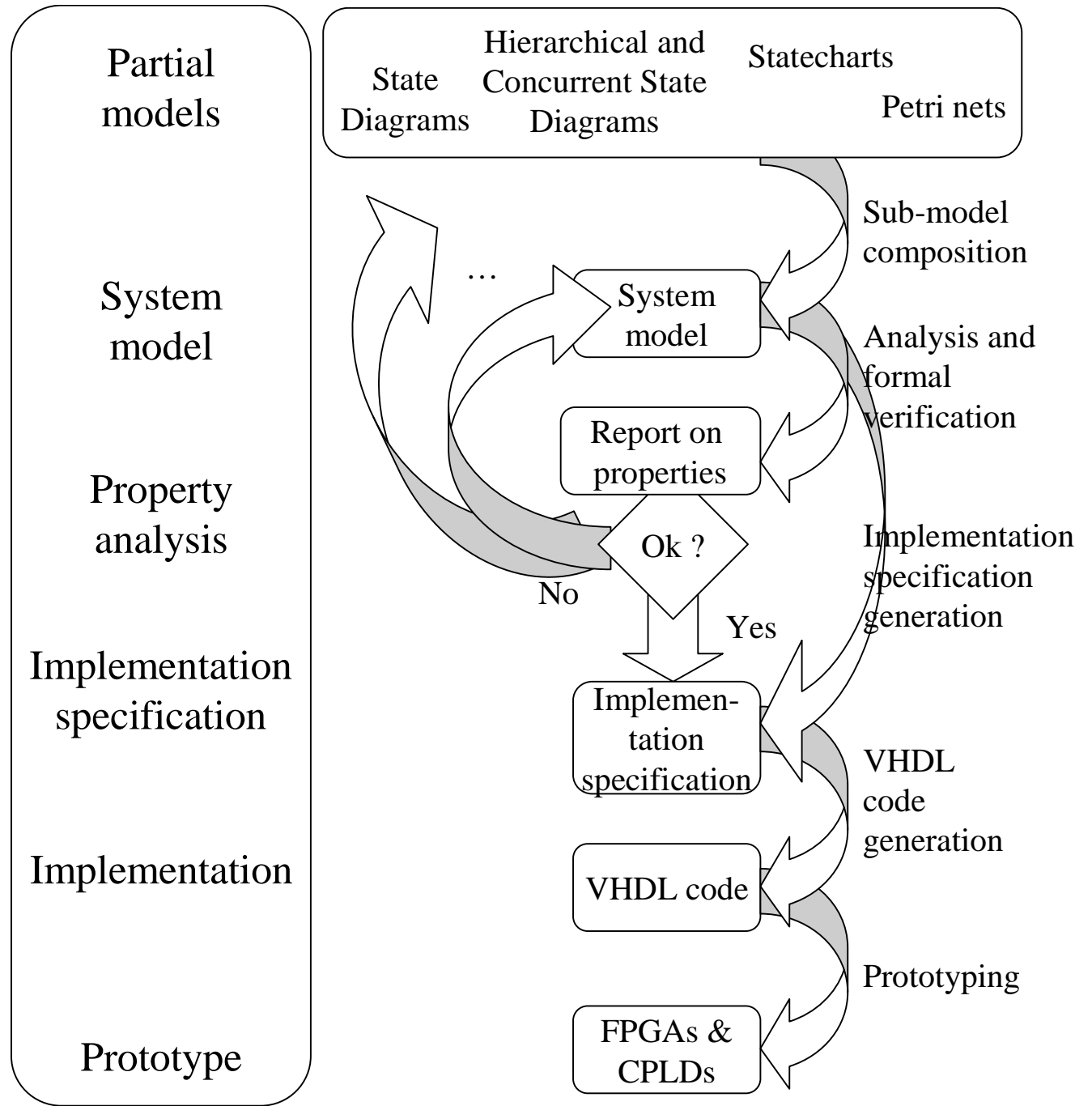
Methodology overview

"The hardware-
software co-design
view"



Methodology overview

"The hardware view"



Starting with Finite State Machines based system modelling

State Diagrams and
State Diagrams with Data Paths (register transfer level)

Foundations:

Based on the modelling of global state and transitions among global states.

Robustness:

Intuitiveness; good for modelling, for implementation specification (executable model) and for documentation.

Weakness:

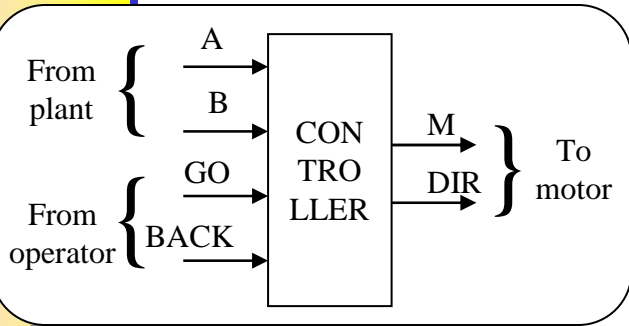
Lack of support for concurrent systems (state space explosion phenomenon);

Lack of hierarchical structuring mechanisms, and folding techniques

Introducing concurrency formalisms

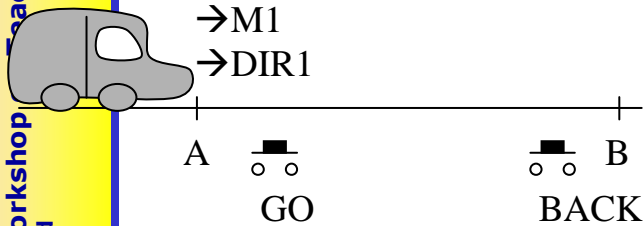
Statecharts = State diagrams extended with hierarchical constructs, orthogonality, and communication mechanism.

Petri nets = Graphical expresiveness,
... complemented by formal
verification capabilities

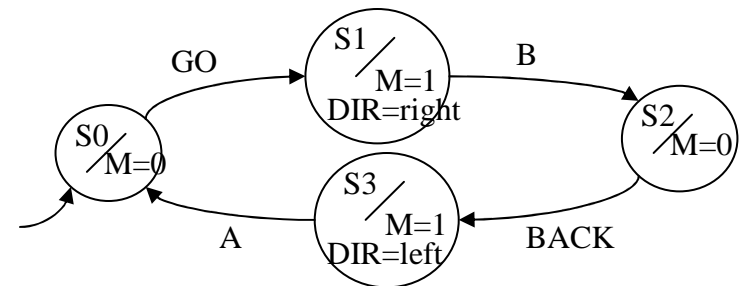


Running example (Ia) State diagram modeling

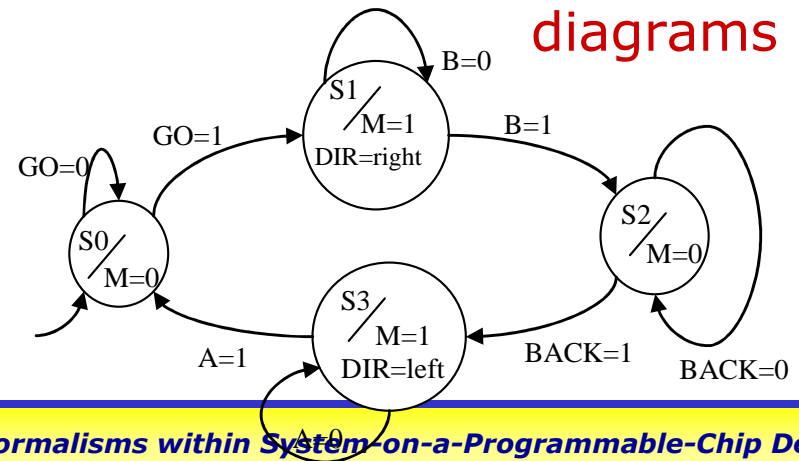
Modeling control of a trolley car for transportation of goods from one point to another



Emphasis on Moore versus Mealy machines, or on signals versus events.

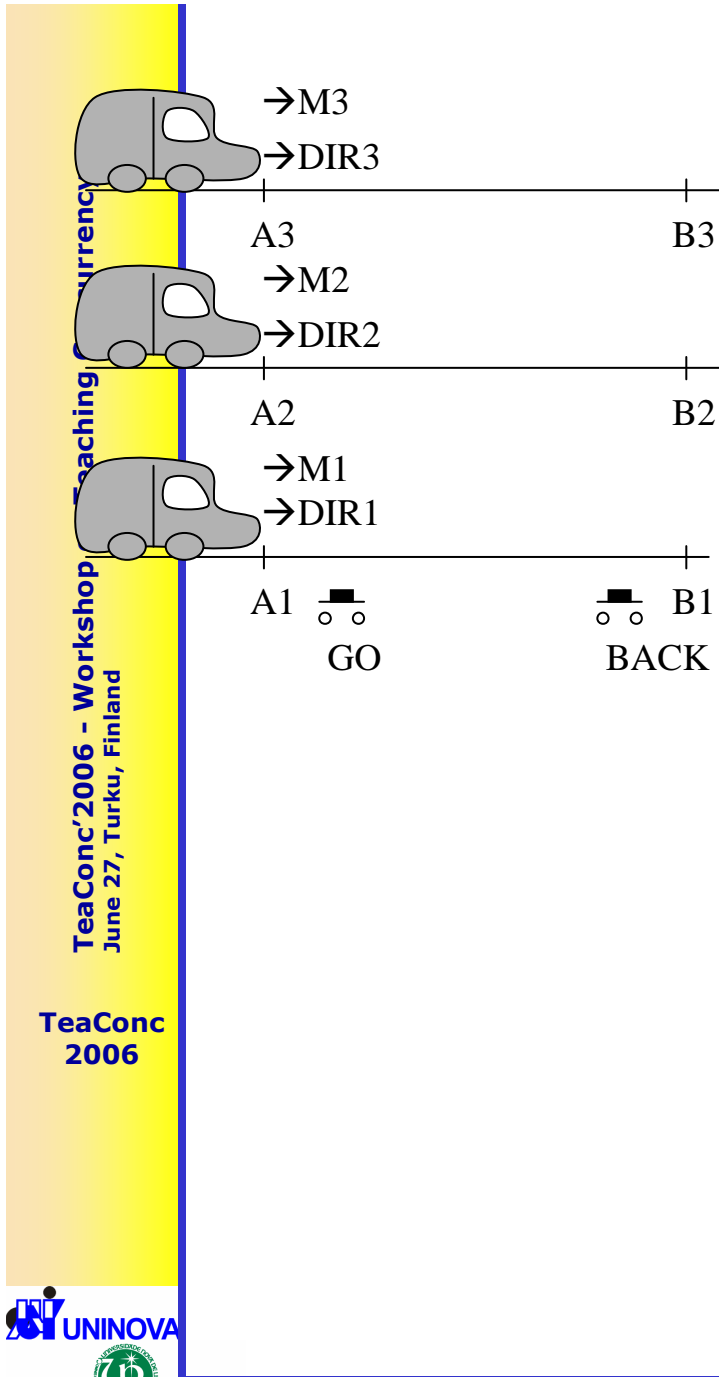


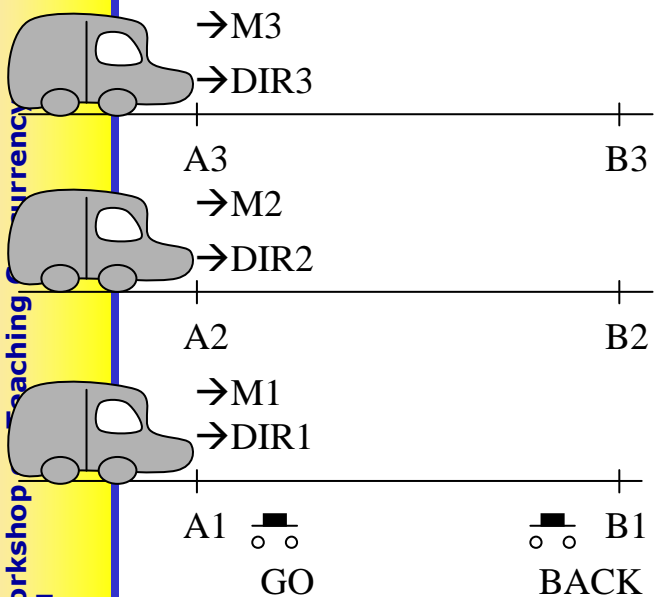
State diagrams



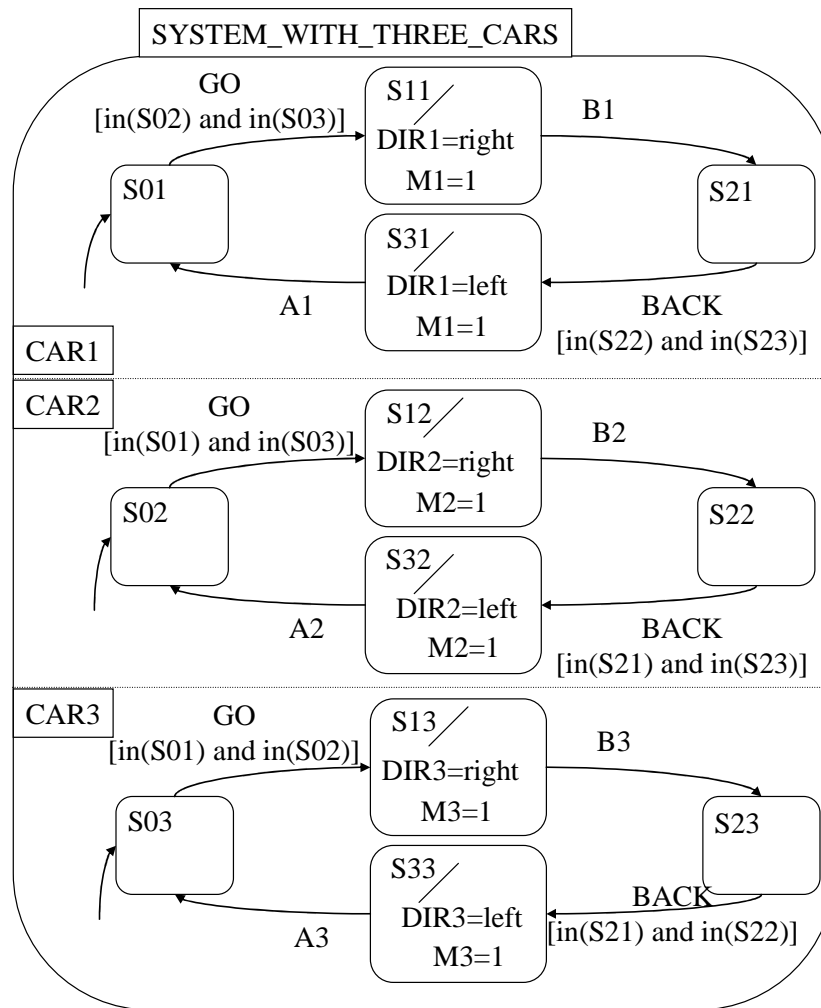
Running example (Ib) Introducing concurrency

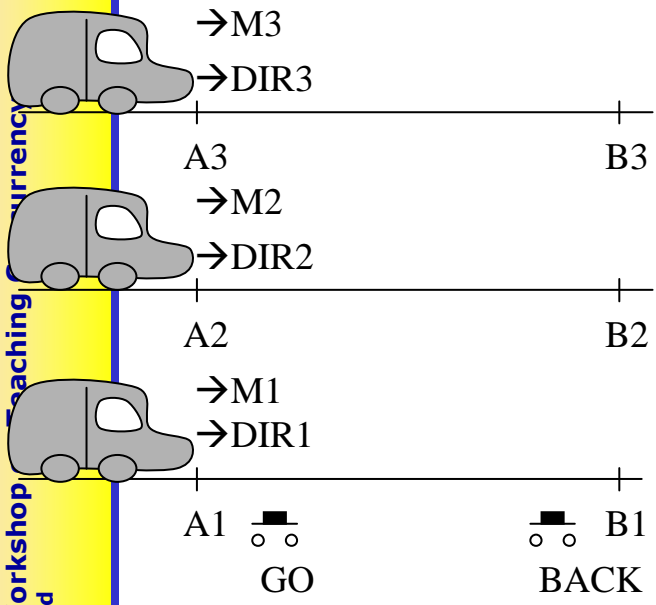
State explosion
problem !!!



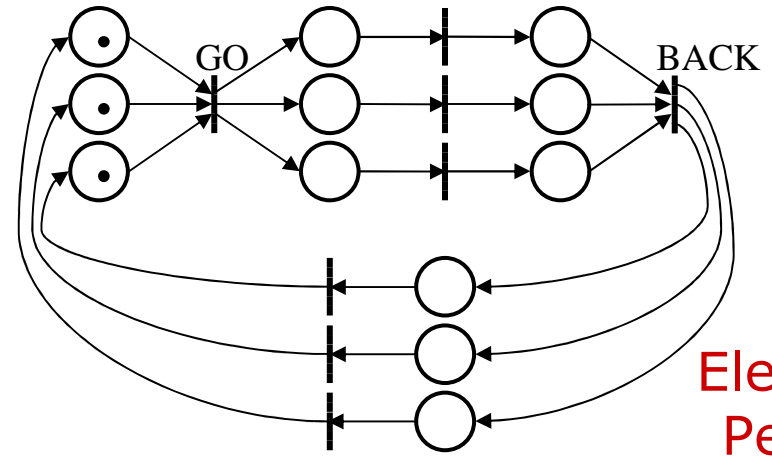


Running example (Ib) Statechart modeling

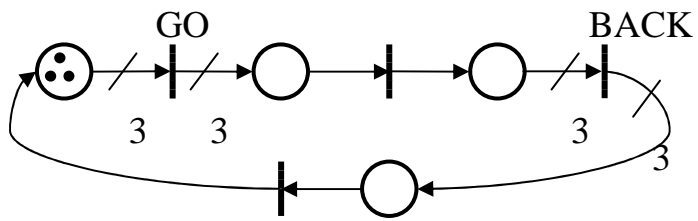




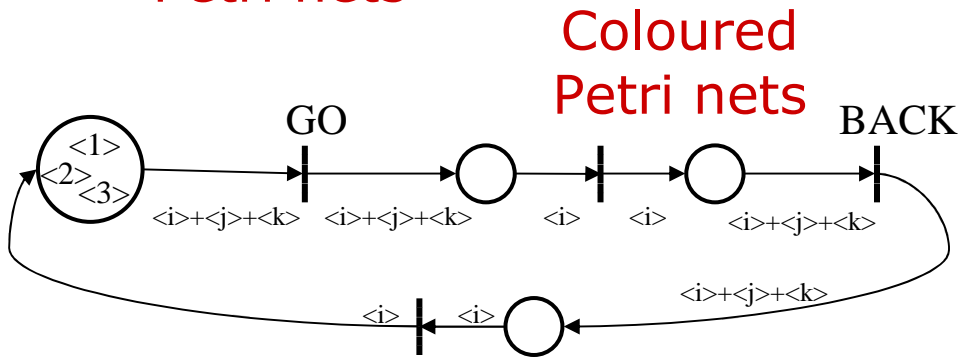
Running example (III) Petri nets modeling



Elementary
Petri nets



Place-transition
Petri nets



Coloured
Petri nets

Laboratory assignments objectives

Goal: Complement theoretical lectures emphasizing laboratory experiments

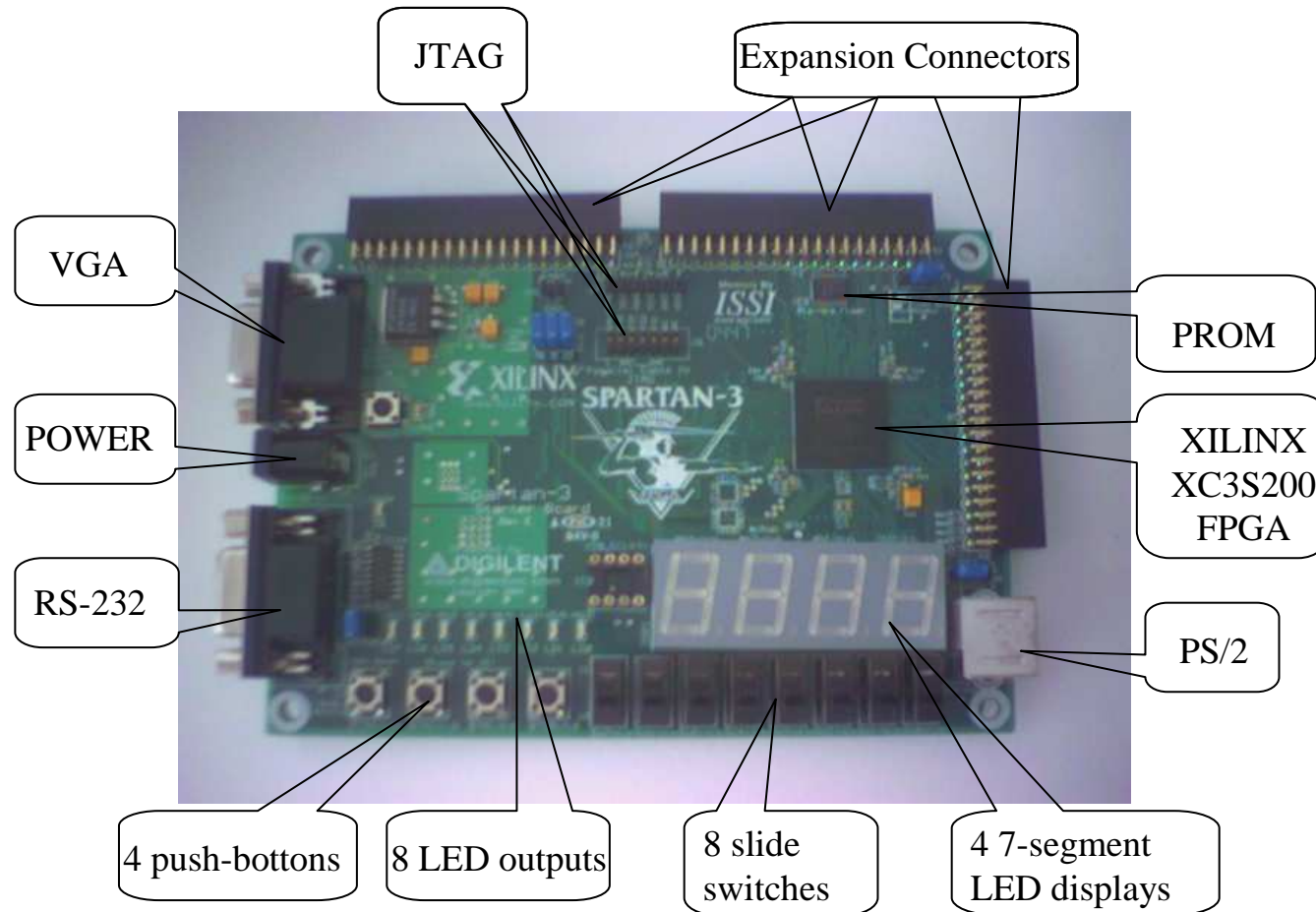
Components potentially presented in lab assignments may include:

- Computer aided engineering component, using a set of engineering software tools (from specific schematic editors to specific compilers and code generators);
- Workbench level instrumentation, to shorten distances and eliminate fears associated with the control of physical devices;
- On system simulation, more and more important along the course (and real life) as the system complexity increases.

Lab assignments plan

- Introductory exercises (from state diagrams to Petri nets)
 - State machine with datapaths
 - Modules (including “on logical equivalency of software and hardware”)
 - Concurrent state diagrams & Statecharts
 - Petri nets
- Mini-project

FPGA-based experimentation kit



Proposed projects

Several families of mini-projects are proposed:

- Having concurrent activity of several entities in the system
- Easy to "tune" to be slightly different for several groups year after year (each group will have a unique project set of requirements)

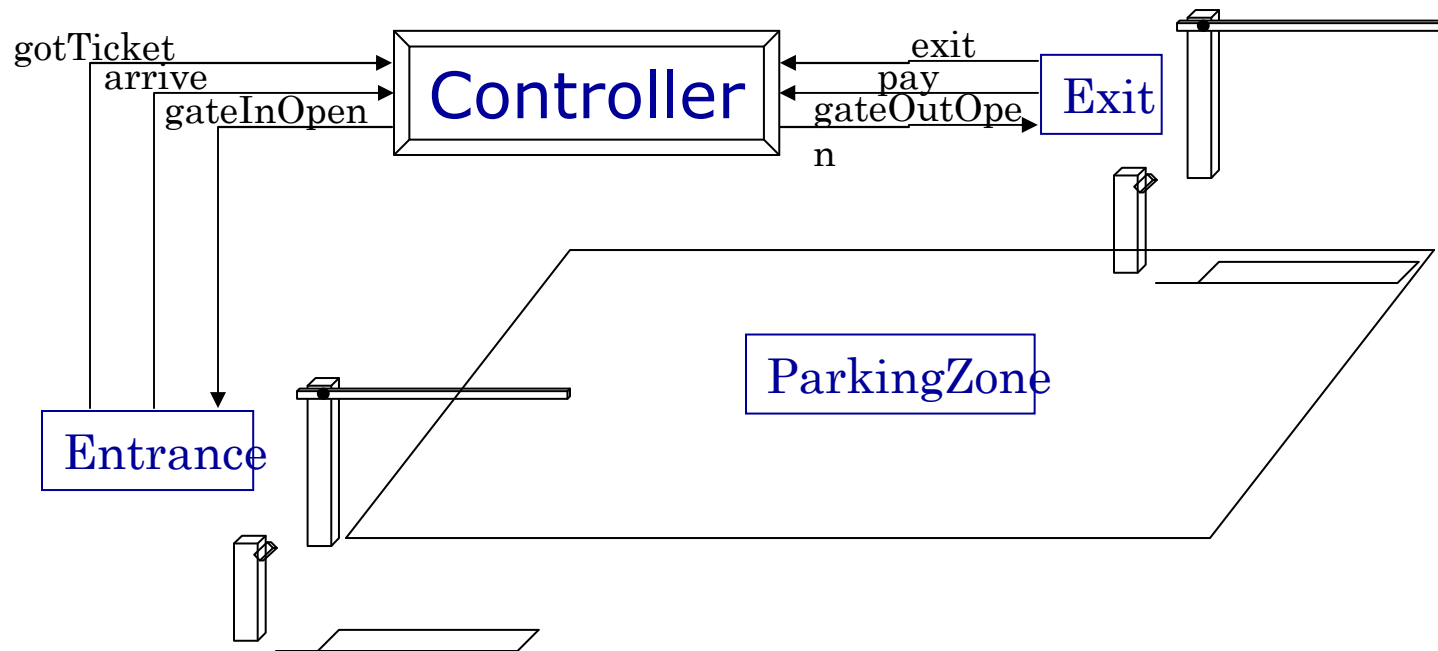
Common steps to all mini-projects:

- modelling through Petri nets, Statecharts or even State Diagrams
- coding in VHDL
- implementation using one FPGA-based didactic board (with one Xilinx Spartan-III FPGA)

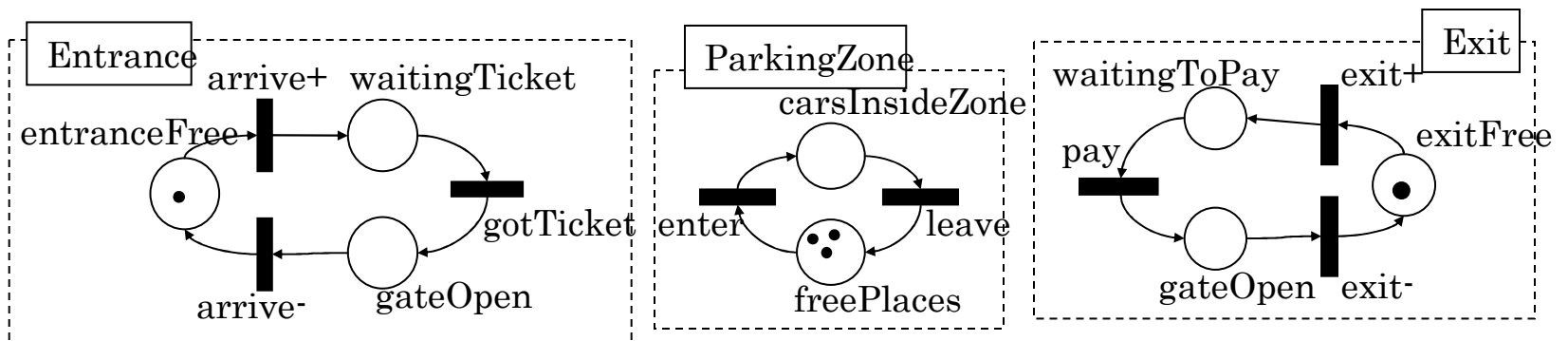
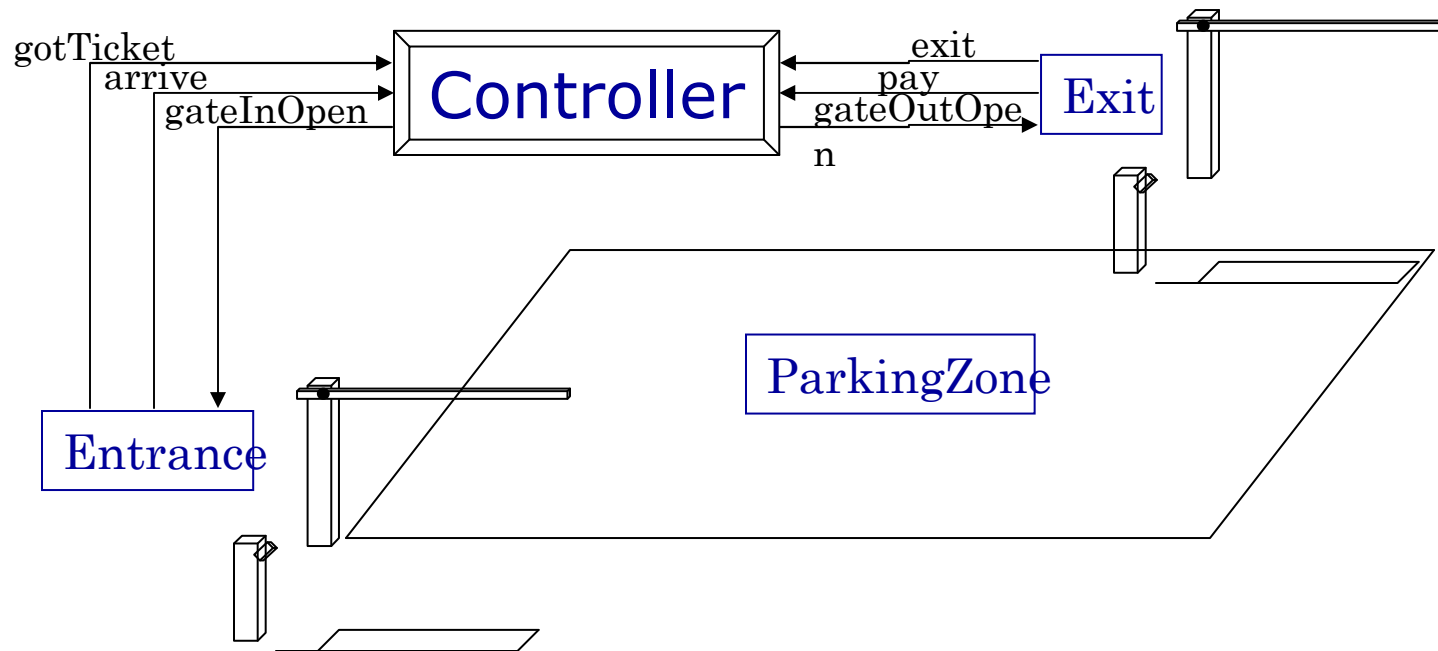
Most popular mini-project:

- parking lot controller

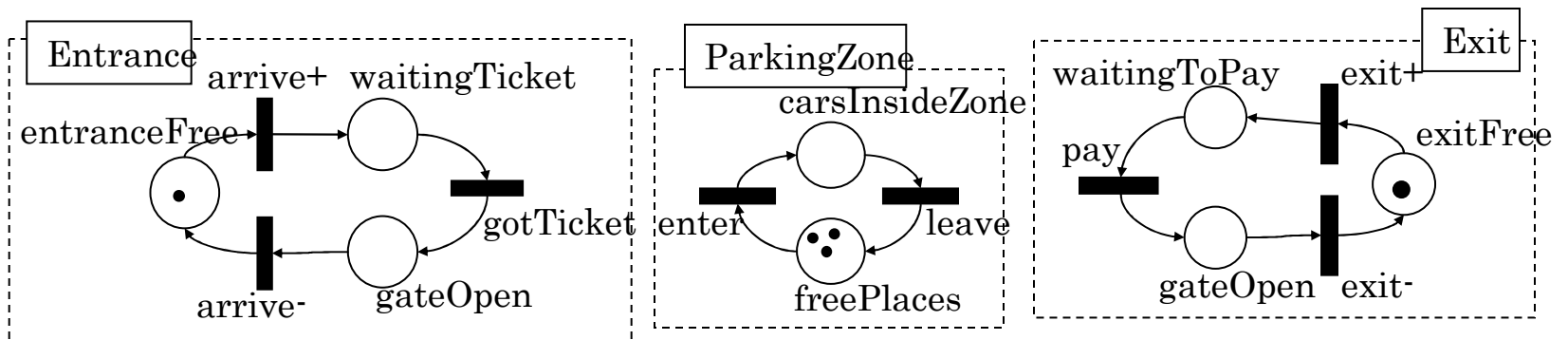
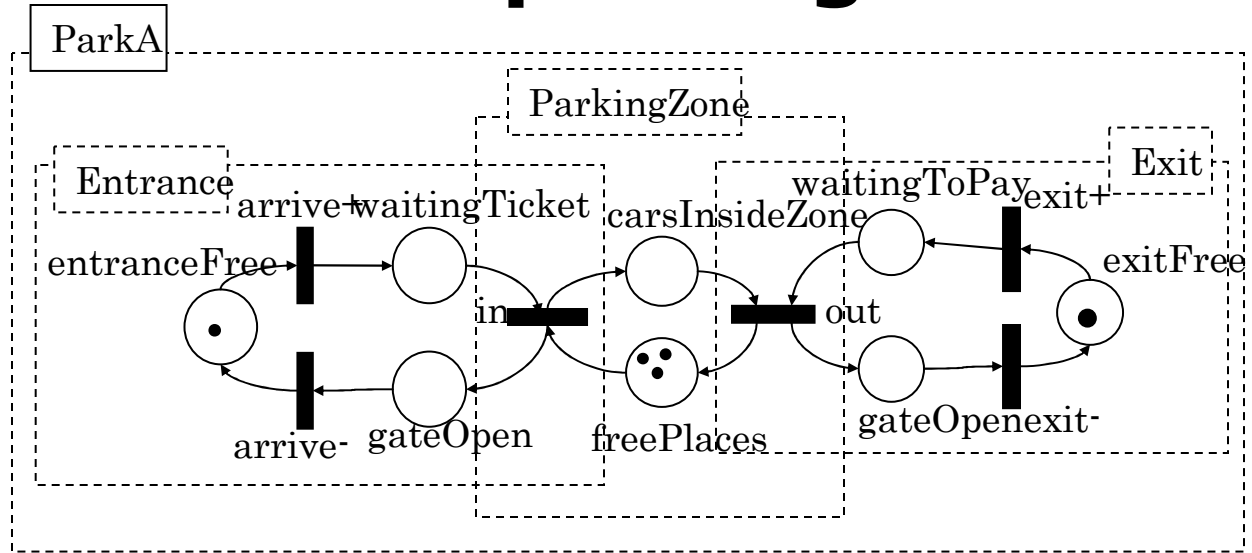
1-in 1-out parking lot - I



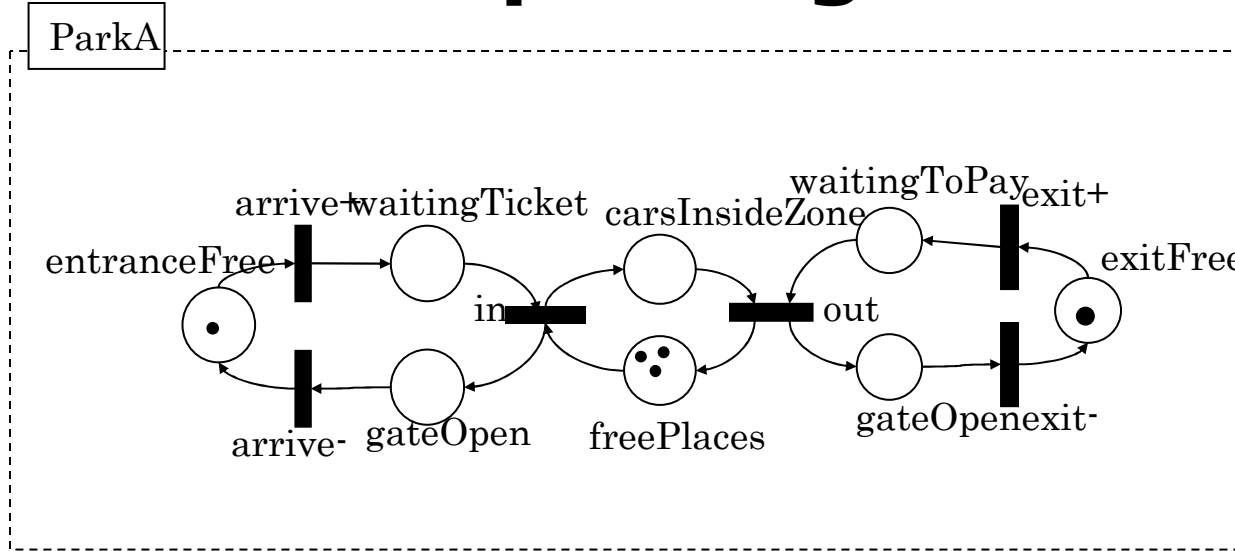
1-in 1-out parking lot - I



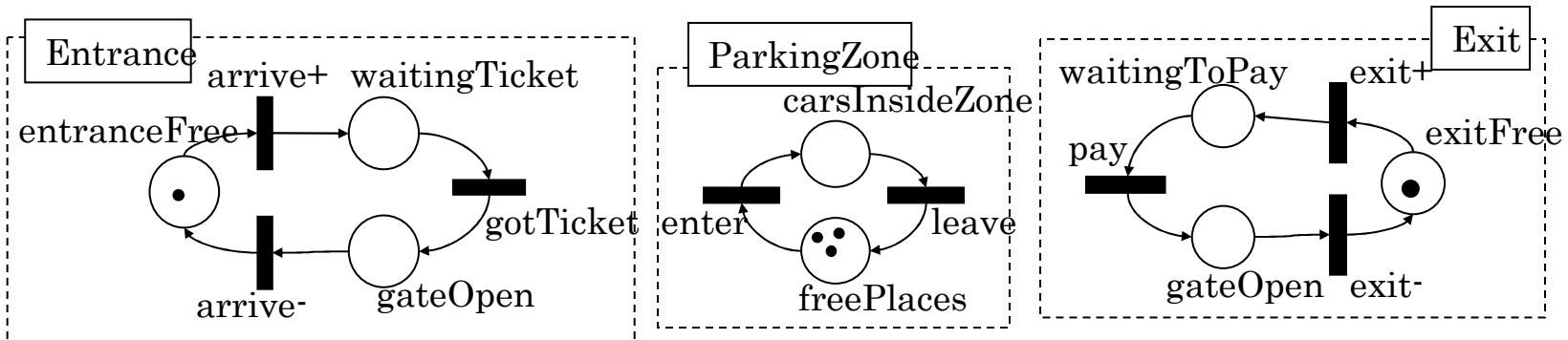
1-in 1-out parking lot - II



1-in 1-out parking lot - II



ParkA = (Entrance + ParkingZone + Exit)
 (gotTicket/enter → in, leave/pay → out)



The car parking lot controller

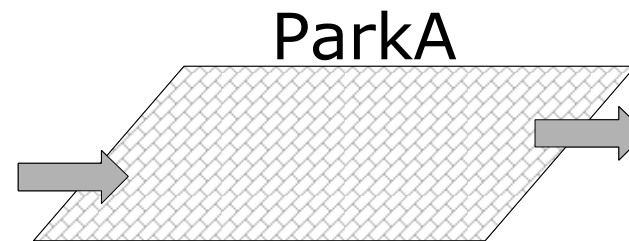
The car parking lot controller proved to be a very flexible application example that can be easily reconfigured keeping complexity at similar levels.

Easy to change requirements at different aspects and levels, namely:

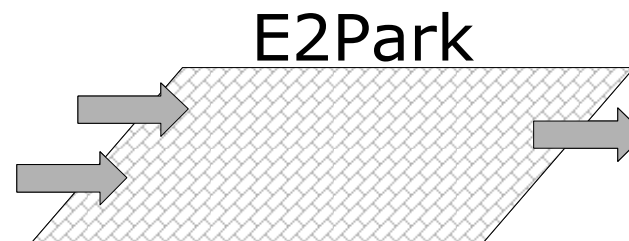
- layout of the parking lot
- robustness of the model

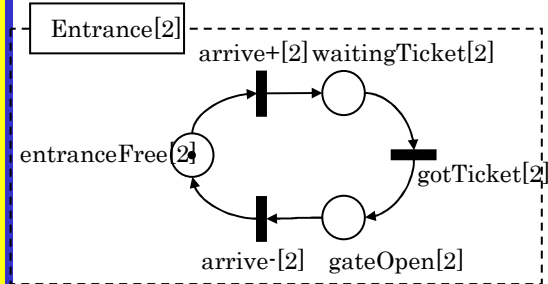
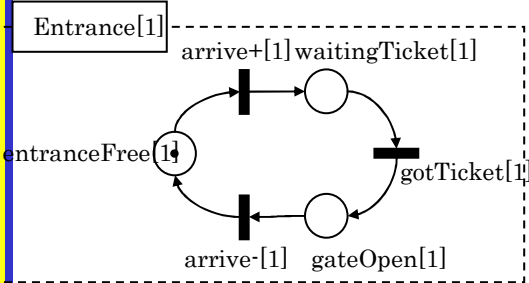
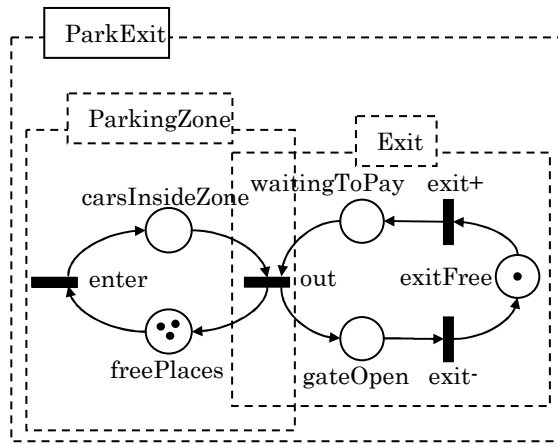
Changing the layout...

From 1-in 1-out

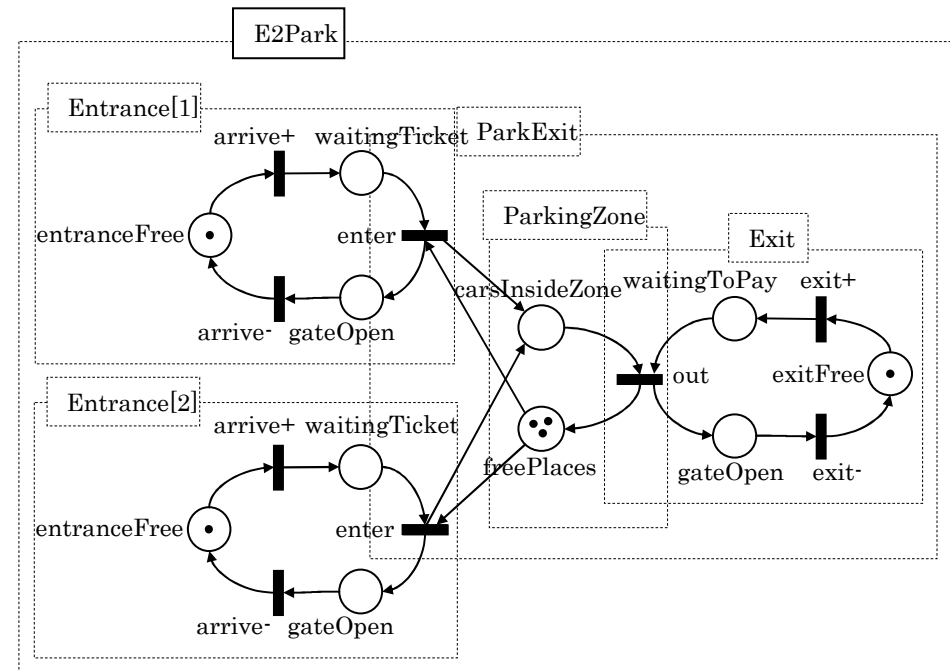


To 2-in 1-out





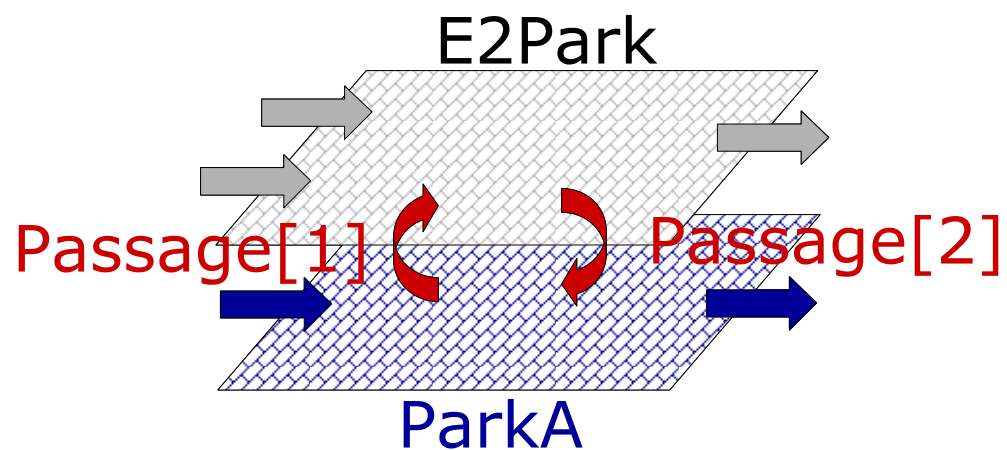
Composition of models



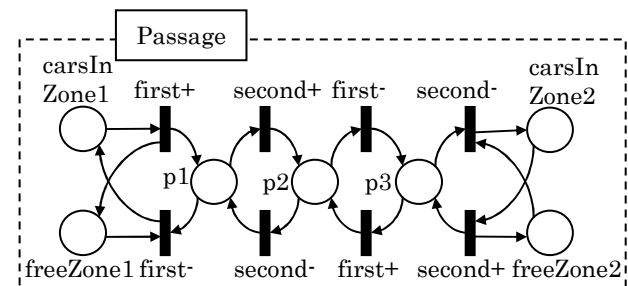
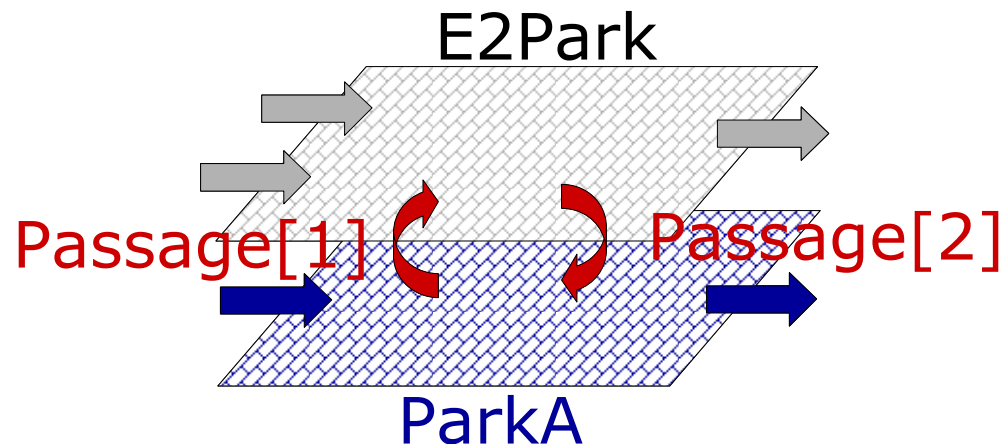
$E2Park = (Entrance[1...2] + ParkExit)$

$((Entrance[i].gotTicket/enter \rightarrow Entrance[i].enter)[i : 1...2])$

Further compositions...



Further compositions...



Adding two passages to ParkA

$$\text{ParkAPassage2} = (\text{ParkA} + \text{Passage}[1\dots2]) \text{ (IC)}$$

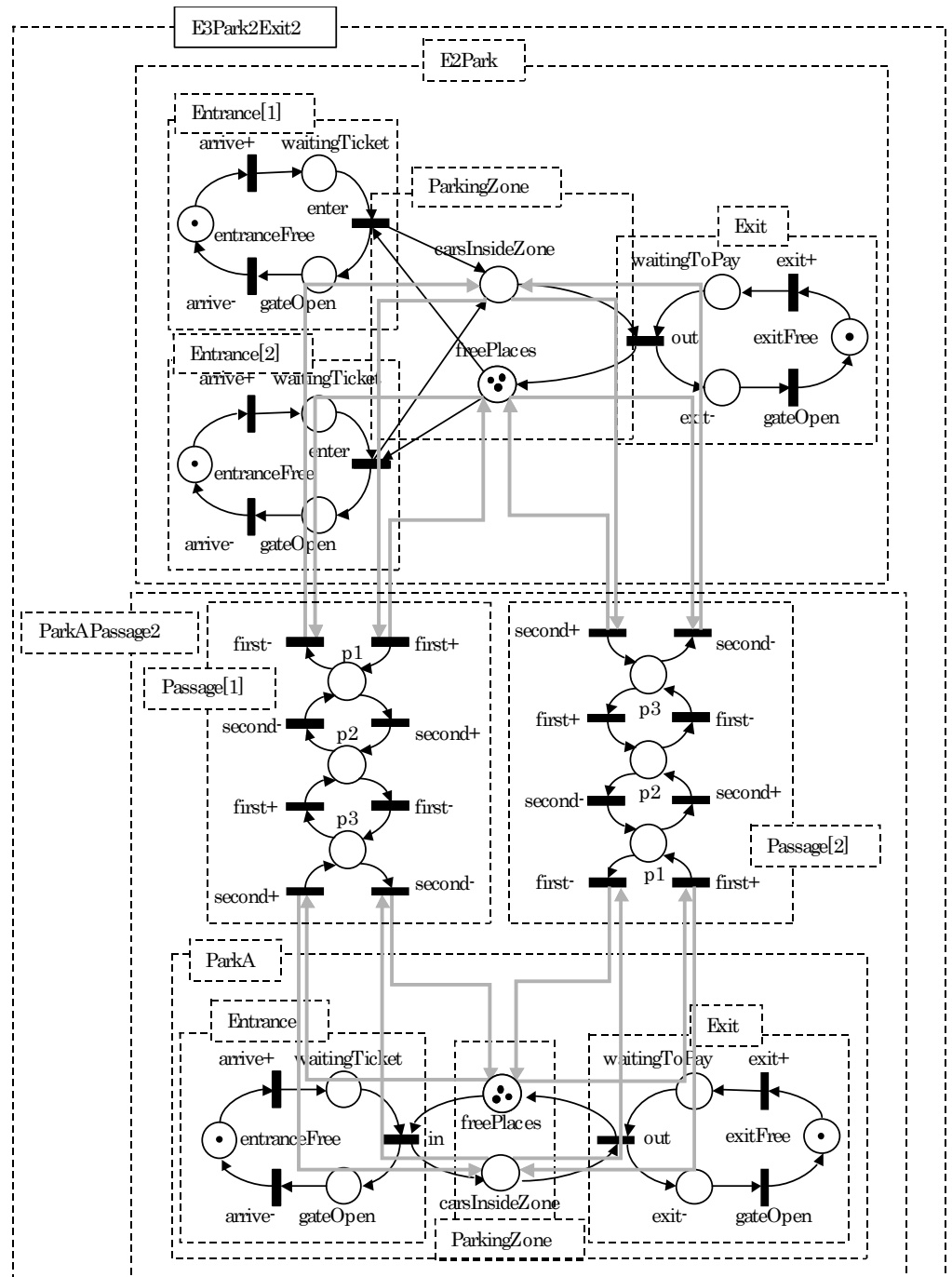
Adding E2Park

$$\text{E3Park2Exit2} = (\text{ParkAPassage2} + \text{E2Park}) \text{ (IC)}$$

Adding a global counter

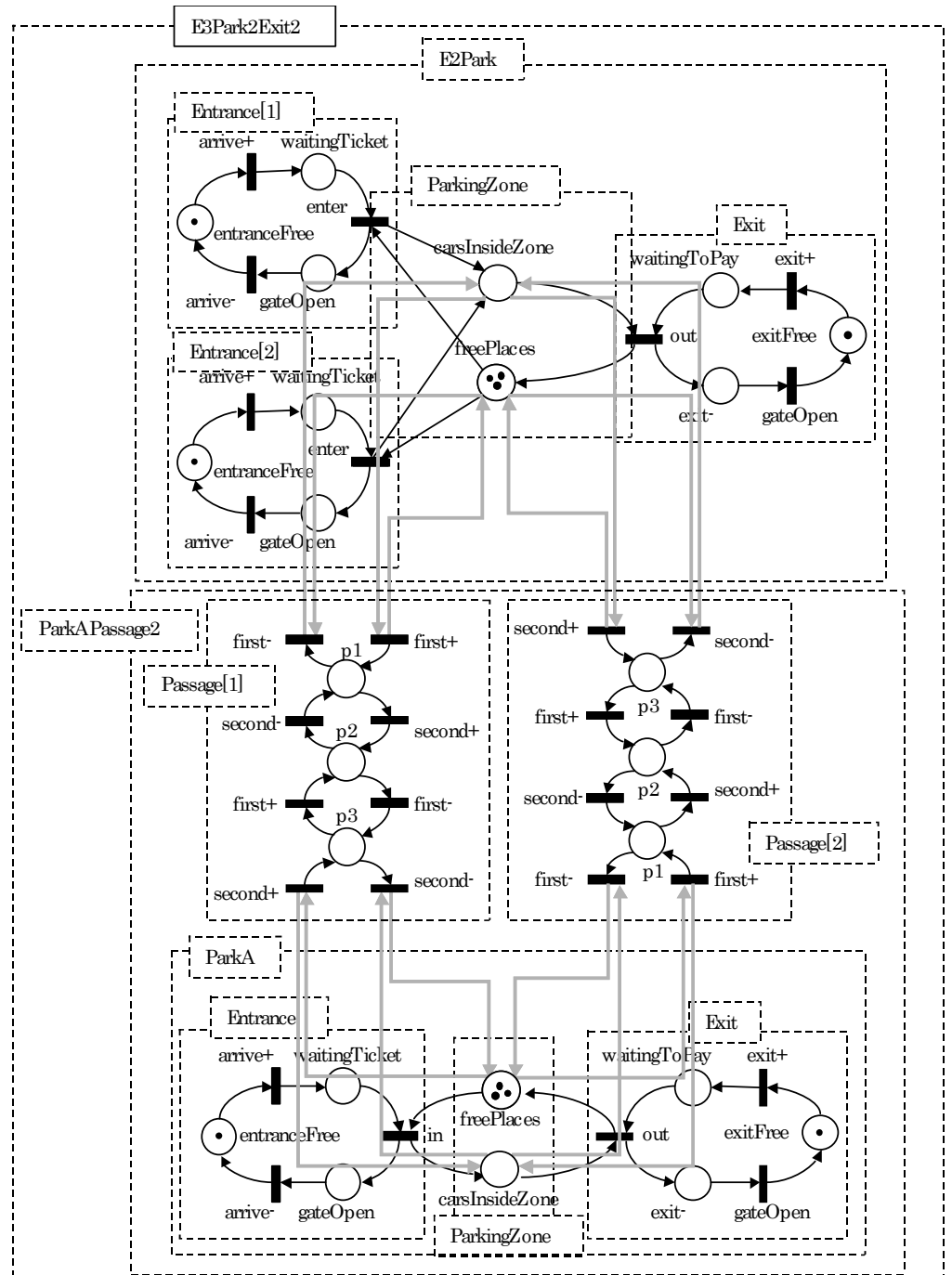
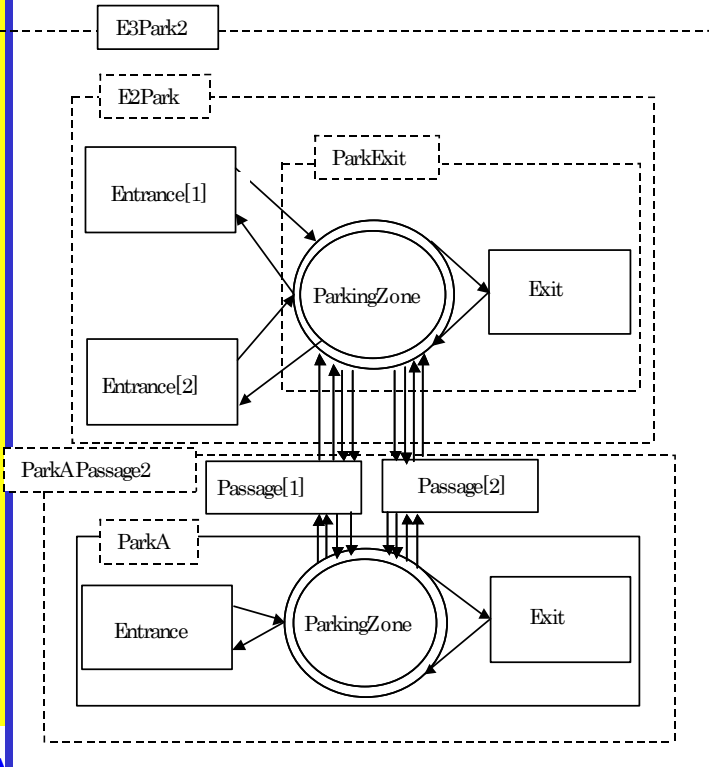
$$\text{E3Park2Exit2Final} = (\text{E3Park2Exit2} + \text{ParkingZone}) \text{ (IC)}$$

Flat representation (simplified model)



Hierarchical versus flat representations

TeaConc
2006



Conclusions

Most of the students enjoyed the proposed lab assignments 😊

One key aspect is availability of experimentation kits (“laboratory anytime anywhere”).

At the end, the students are forced to choose one modeling formalism for their project, according with system’s requirements.

Several key concepts for embedded systems design were exercised, from theoretical to implementation issues: concurrency modeling, model composability, reusability, hardware description languages, reconfigurable devices, ...

Using Concurrency Modeling Formalisms within System-on-a-Programmable-Chip Design

**Luís Gomes,
Anikó Costa,**

lugo@uninova.pt
akc@uninova.pt



Thank you for attention

LISBON - PORTUGAL