Design of Complex Embedded Systems Based on Different Petri Net Interpretations

Prof. Dr.-Ing. habil.

Wolfgang Fengler

Technical University of Ilmenau

Faculty of Computer Science and Automation

Institute for Theoretical and Technical Informatics

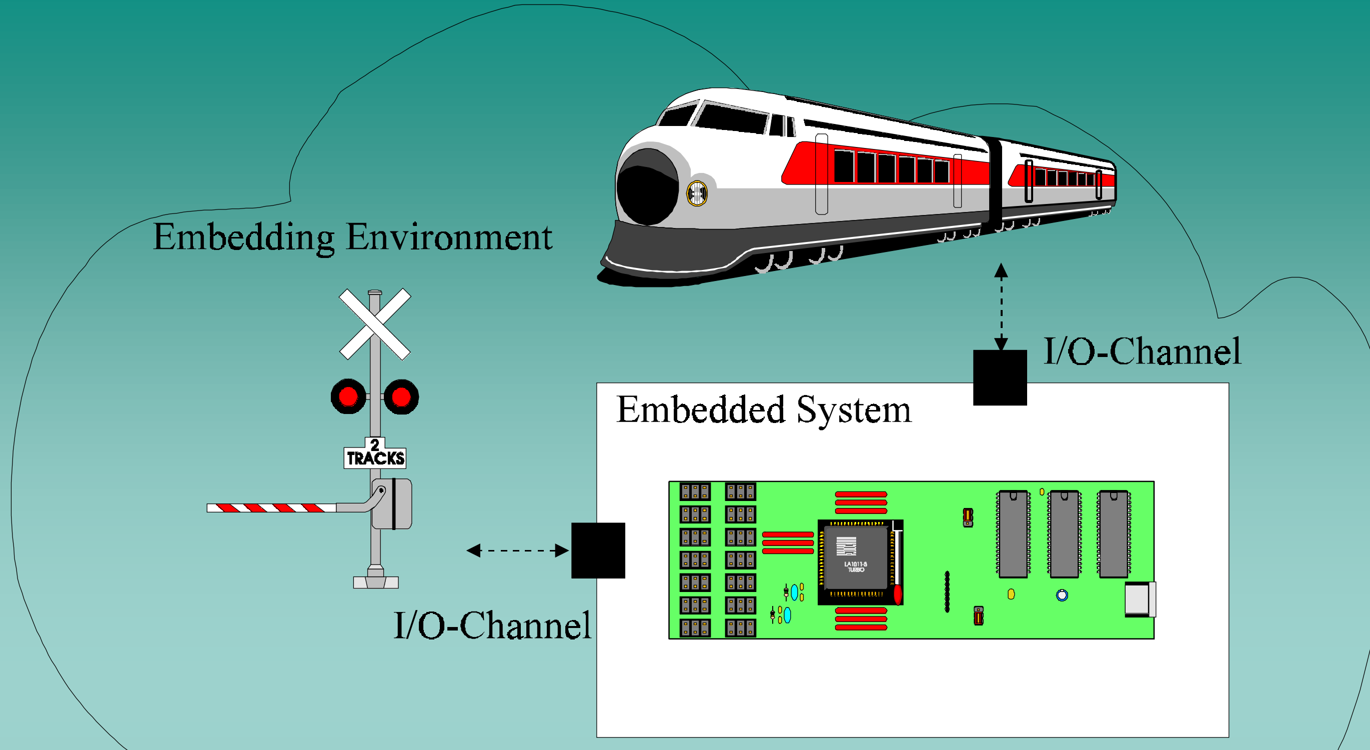
Department of Computer Architecture

(Computer Architecture and Parallel Systems)

e-mail:wfengler@theoinf.tu-ilmenau.de

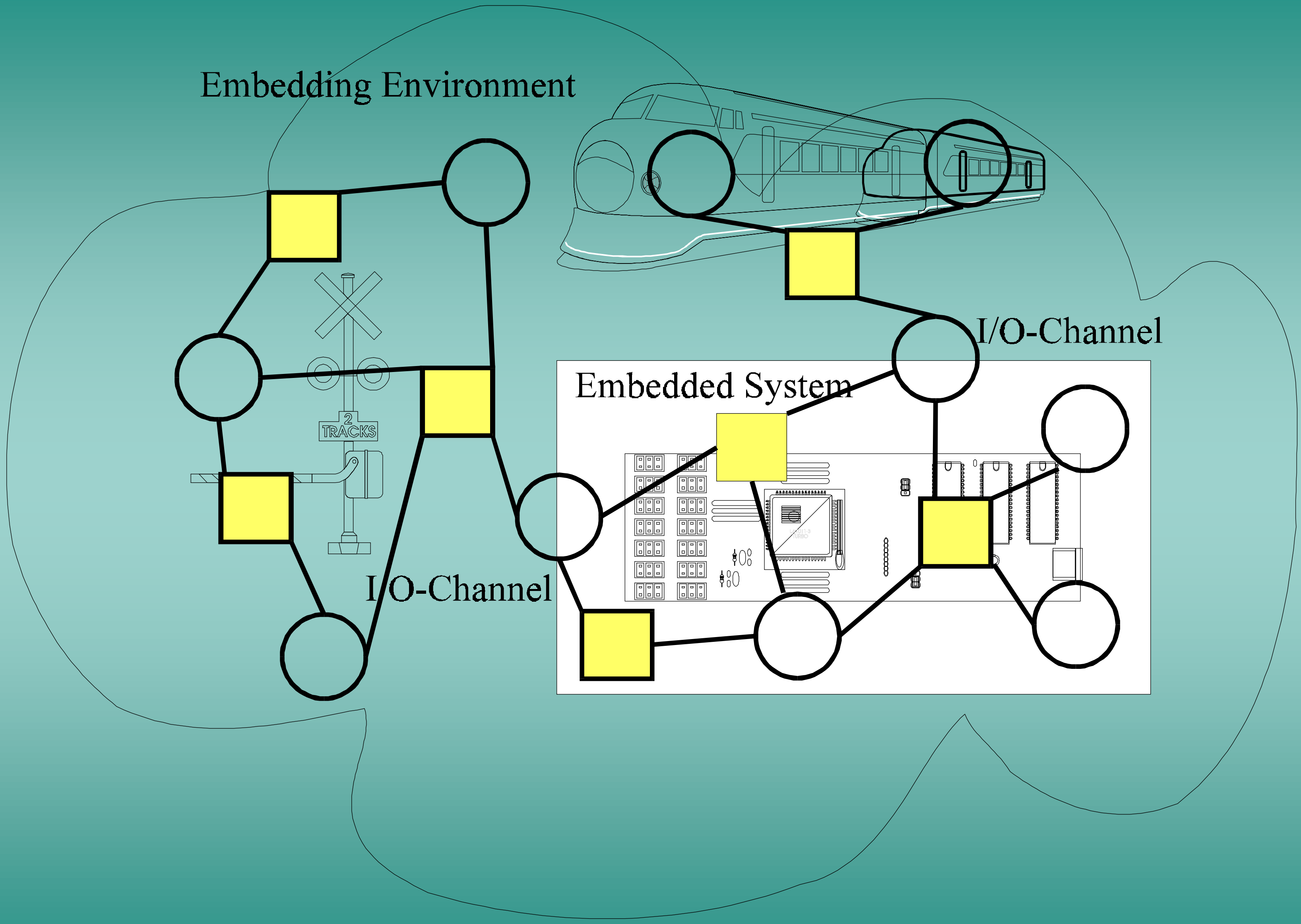
Topics

1. Introduction
2. Object Oriented Design
3. Distribution Procedure
4. Communication Design
5. Hardware Design
6. Analysis Methods
7. Summary

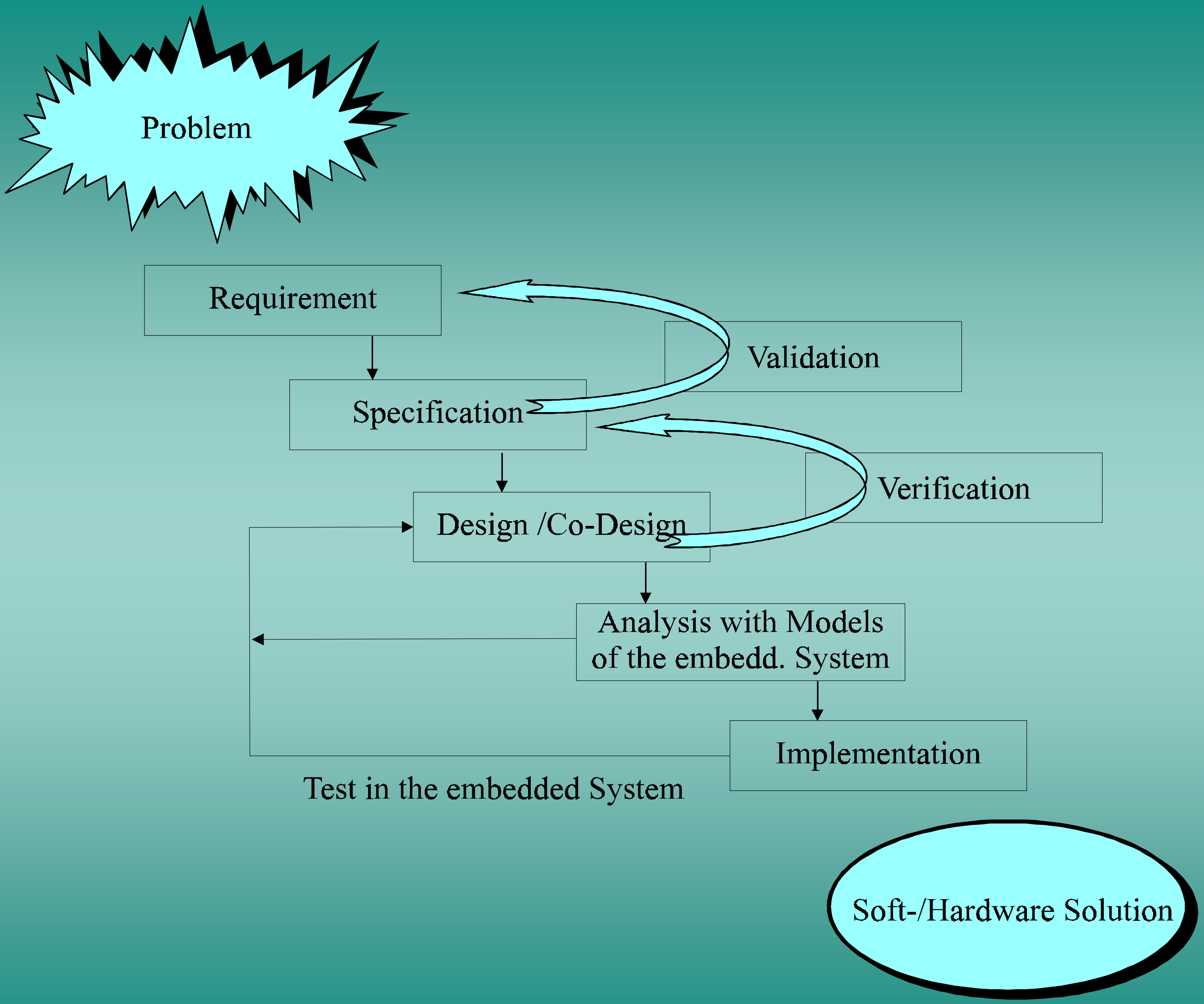


- ➤ **Real-Time Capability**
- ➤ **Communication**
- ➤ **Safety-Relevant Aspects**

Parallel High-Performance-Architecture

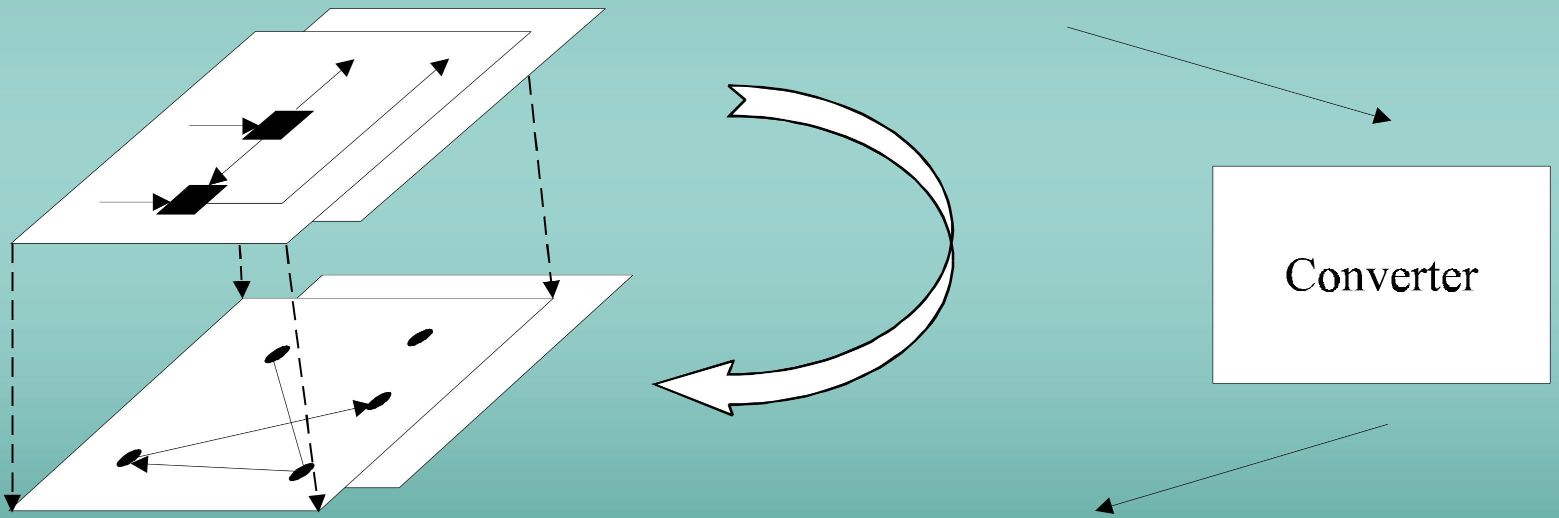

Embedding Environment
I/O-Channel
Embedded System
2 TRACKS
I/O-Channel

1


Problem
Requirement
Validation
Specification
Verification
Design /Co-Design
Analysis with Models of the embedd. System
Implementation
Test in the embedded System
Soft-/Hardware Solution

Problem Description

Description Notation



Basis Notation

Topics

1. Introduction
2. <u>Object Oriented Design</u>
3. Distribution Procedure
4. Communication Design
5. Hardware Design
6. Analysis Methods
7. Summary

Developed OO - Models

Object-Process-Model

- PN based method for OO process modelling
- Generation of source code e.g. for Java or C++
- Formal verification by transformation into coloured Petri Nets

For Computer Scientists

Object Nets

- OO design model with underlying PN interpretation
- Code generation for assembler and high-level-languages
- On-the-fly PN generation for simulation und analysis

For Engineers

Object Nets

Object Oriented Design Model
Based on the Petri Net Theory

Combination of OO-Paradigm with PN-Theory

- Inheritance and reuse
- Close to real-world objects
- Polymorphic specifications

- Simulation and analysis
- Verification and formal description
- Concurrency and real-time behaviour

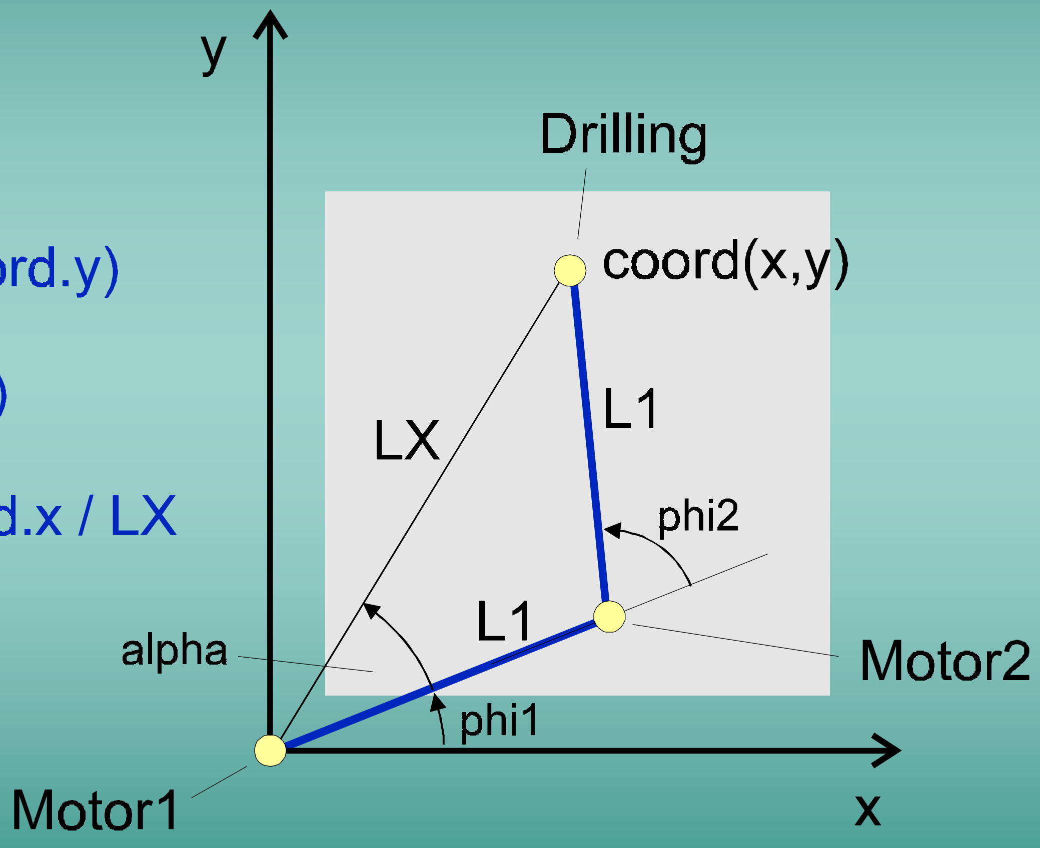
Controlling two Angles for Drilling

LX = hypot (coord.x, coord.y)

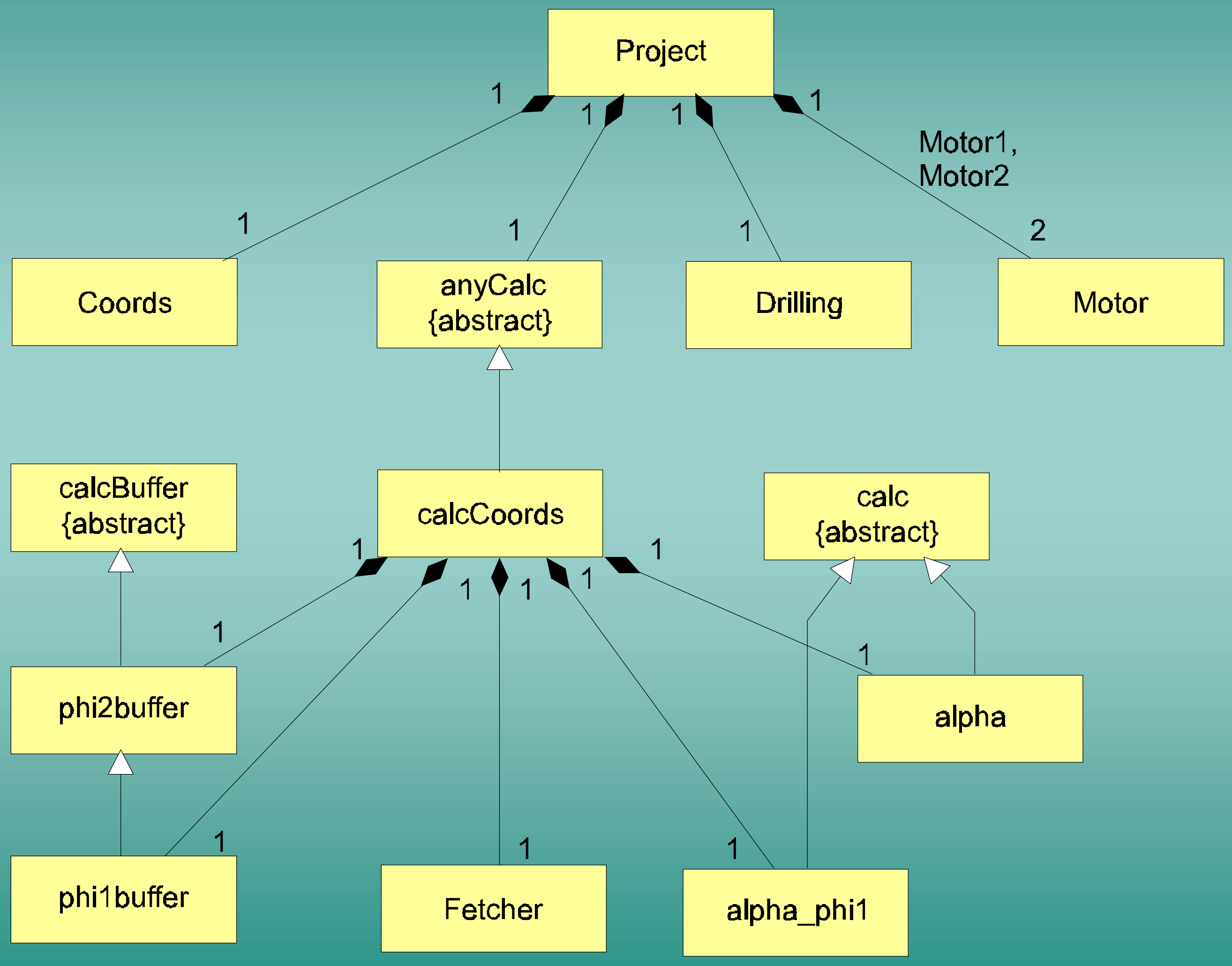
cos (alpha) = LX / (2*L1)

cos (alpha+phi1) = coord.x / LX

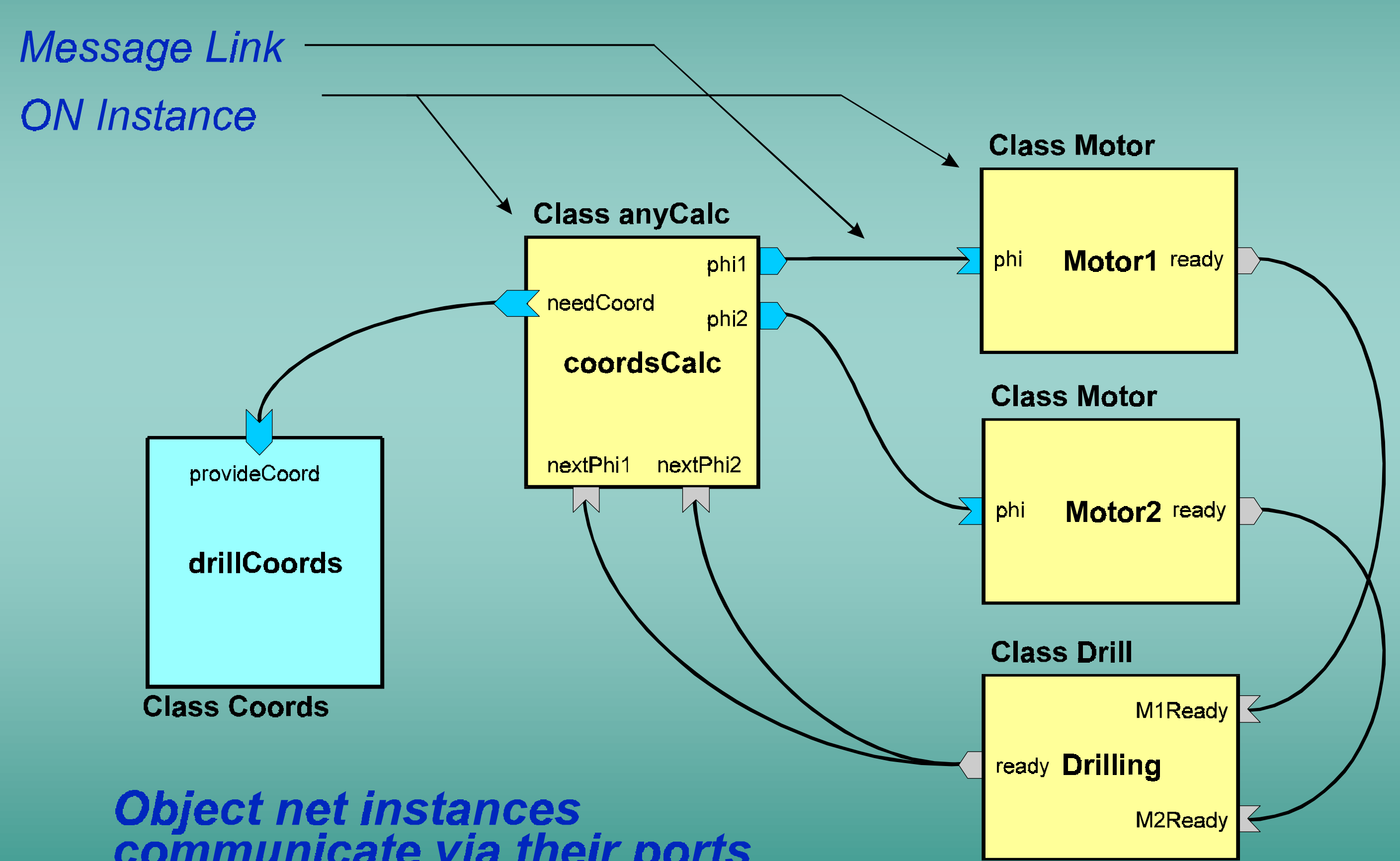
phi2 = 2 * alpha



UML Static Structure Diagram



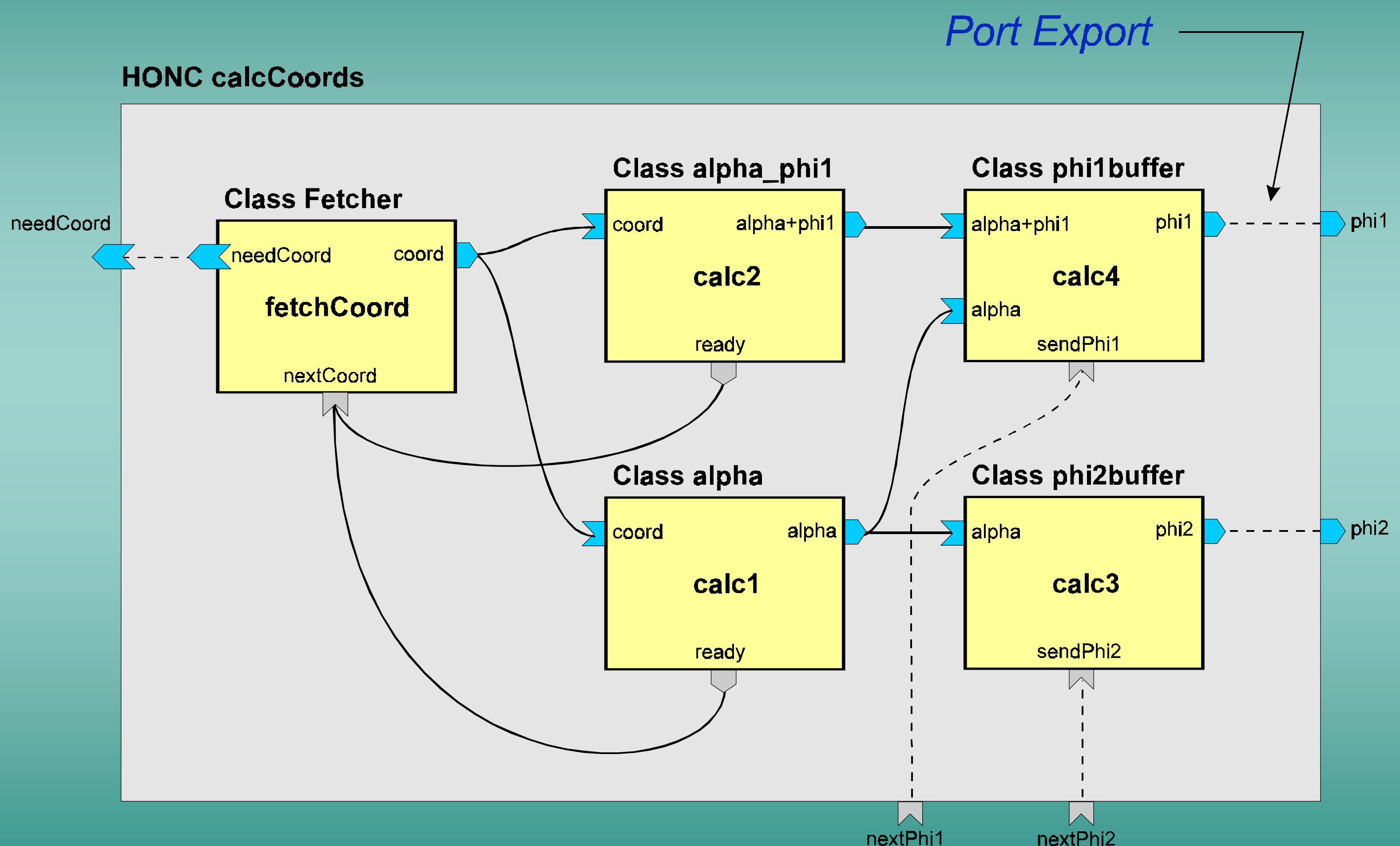
The Object Net of the Drilling Project
(Collaboration Diagram)



Object net instances communicate via their ports

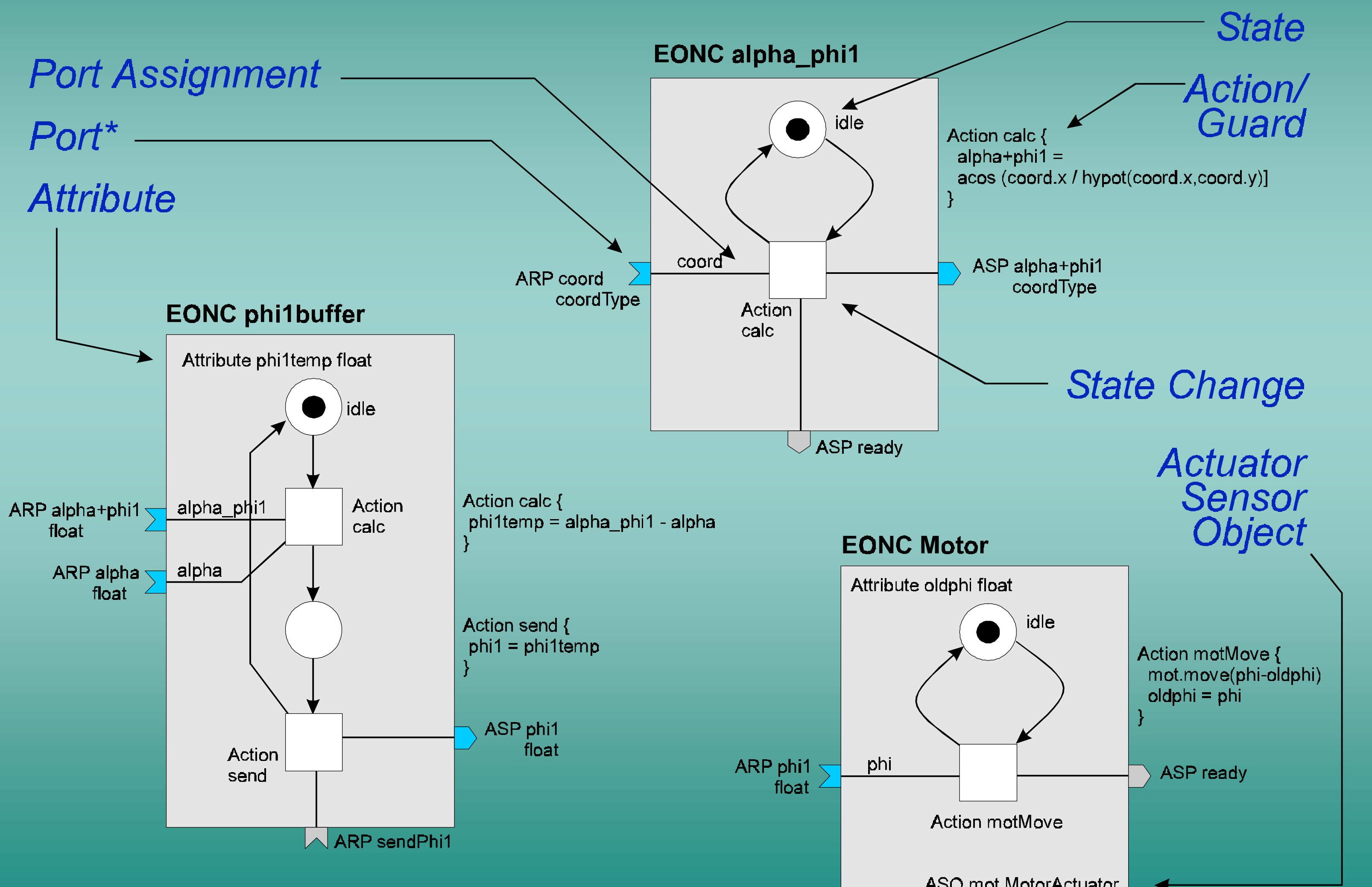
Message Links connect them each other

HONC - Hierachical Object Net Class



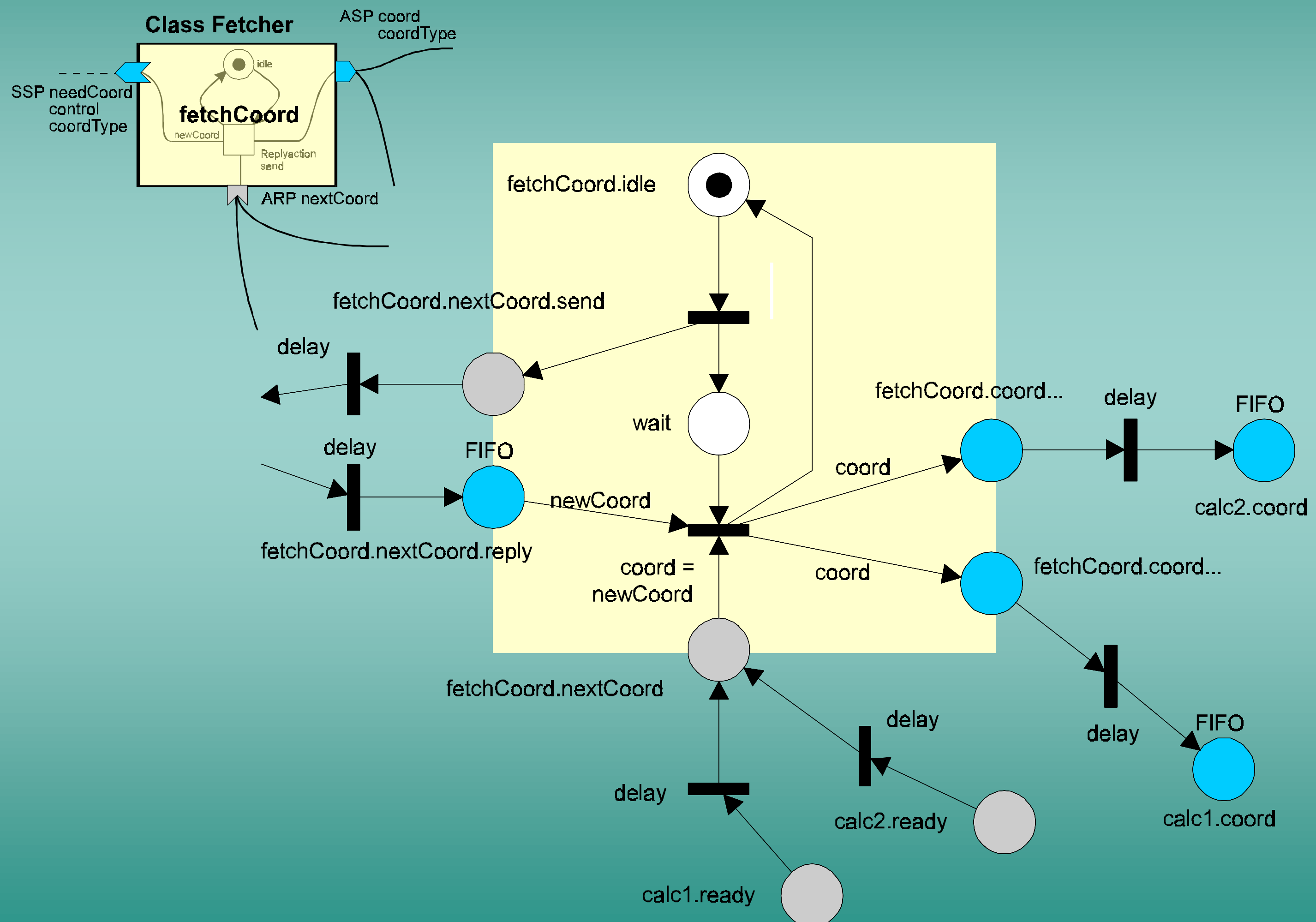
2

EONC - Elementary Object Net Classes
(Buffered State Machines)



* ASP, ARP, SSP, SRP, PP

The Corresponding Petri Net



Object Process Model (OPM)

Petri Net Based Method for Object Oriented Process Modelling

Statical Aspects

- Class
- Role

Class Role Model + **OPM**

(Static Structure Diagram)

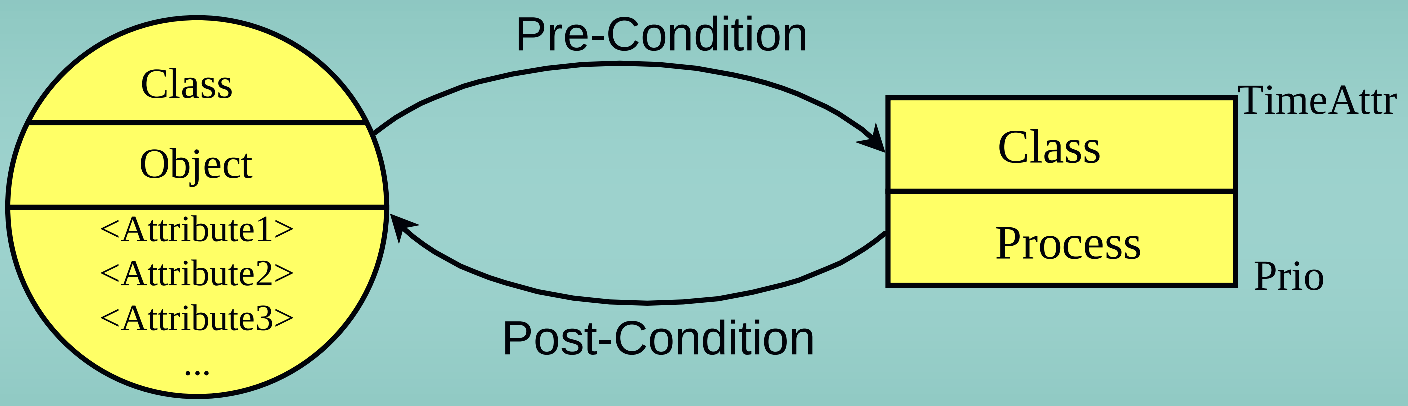
Dynamical Aspects

- Object
- Process
- Pre- and Post-Conditions

Formal Verification by Transformation into Coloured Petri Nets

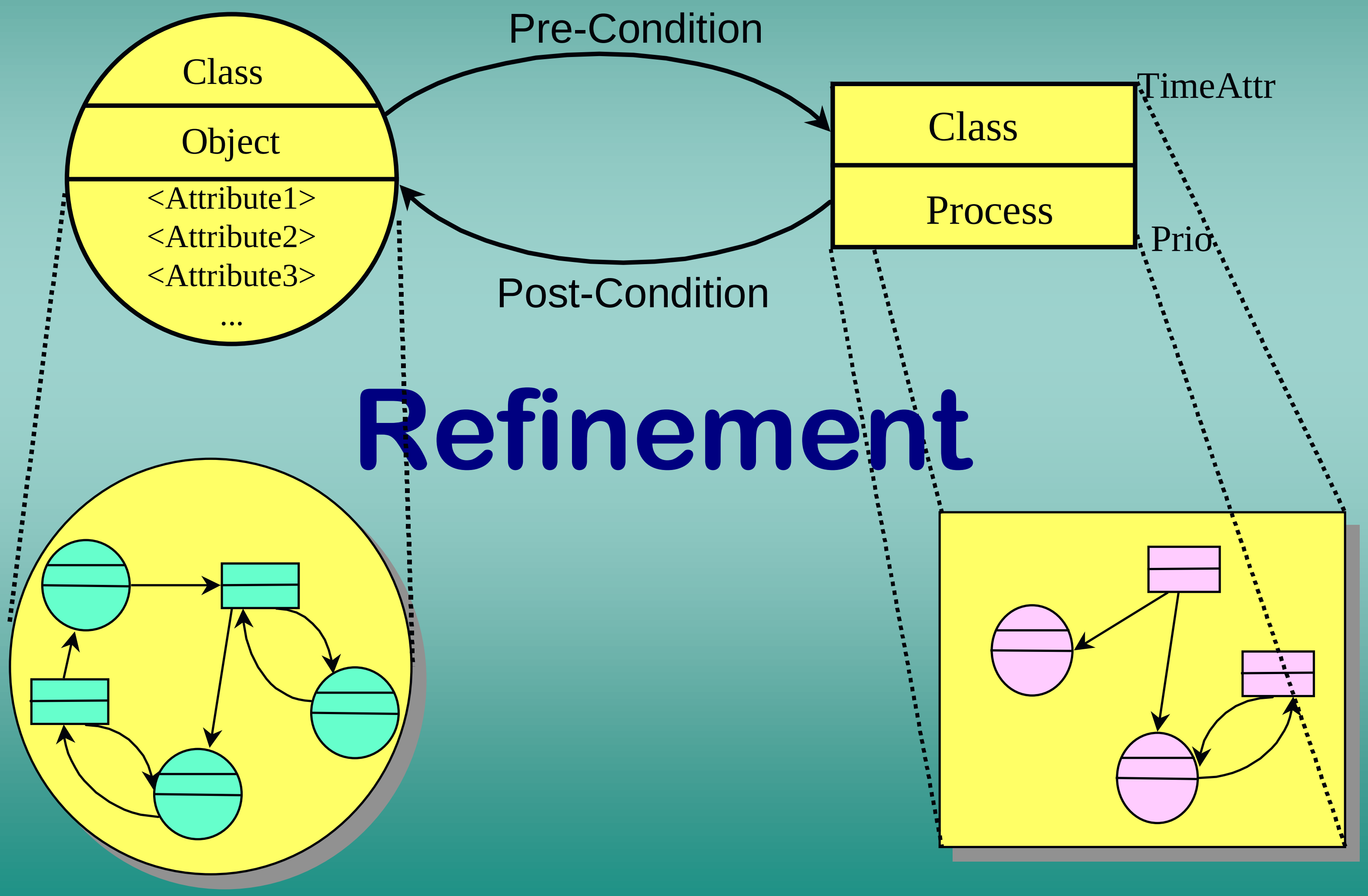
Graphical Notation of an OPM

Object Process

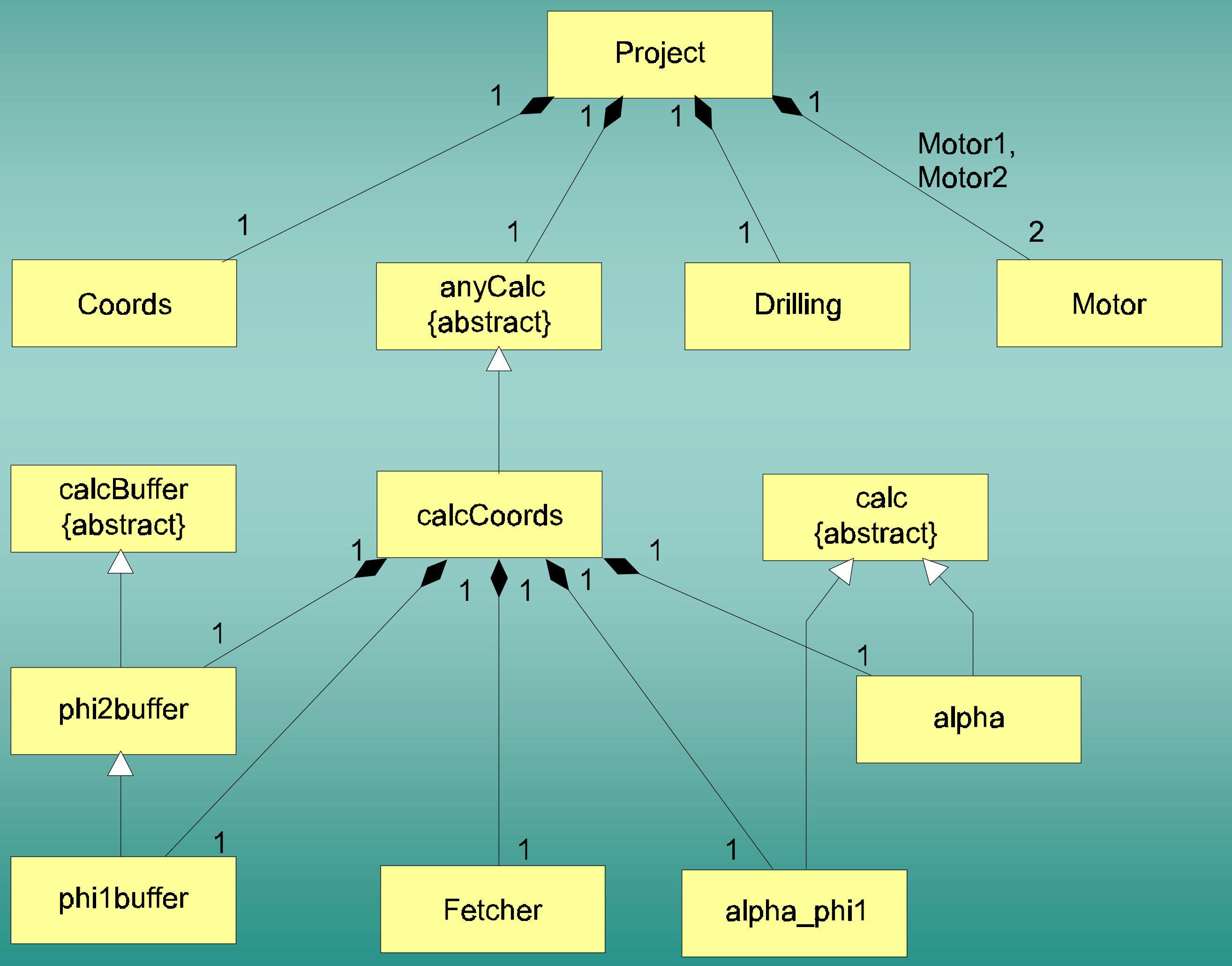


Directed arcs labelled with terms built by attributes and operators

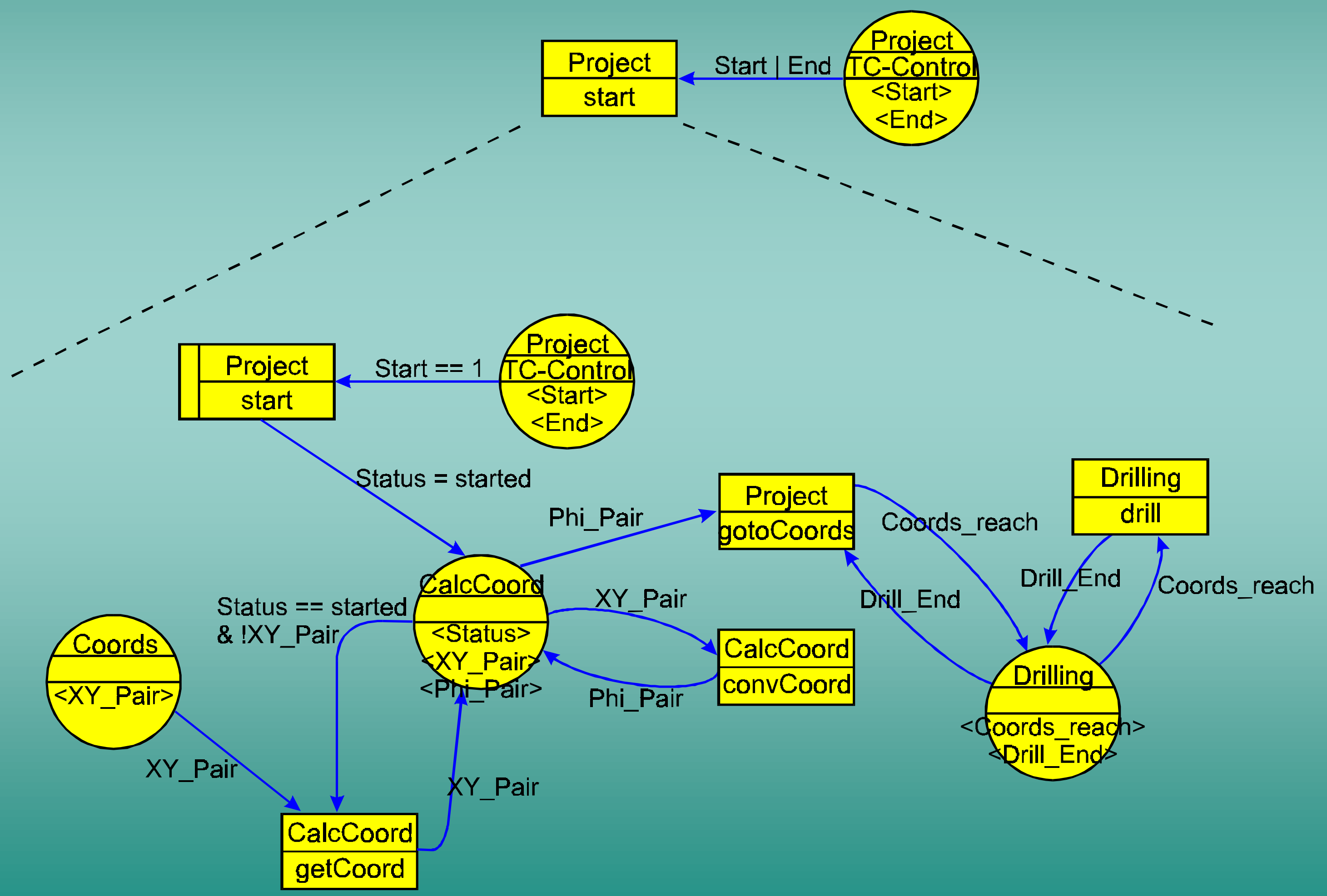
Using Hierarchy in the OPM



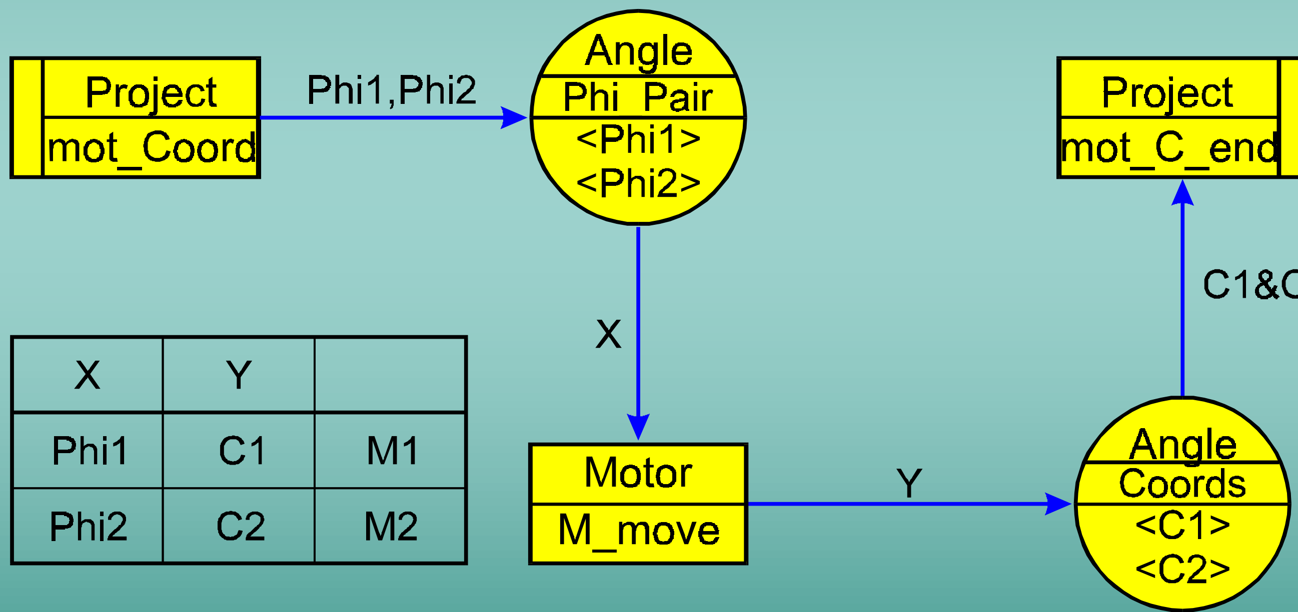
UML Static Structure Diagram



The Main Process "start" and its Refinement (partly)

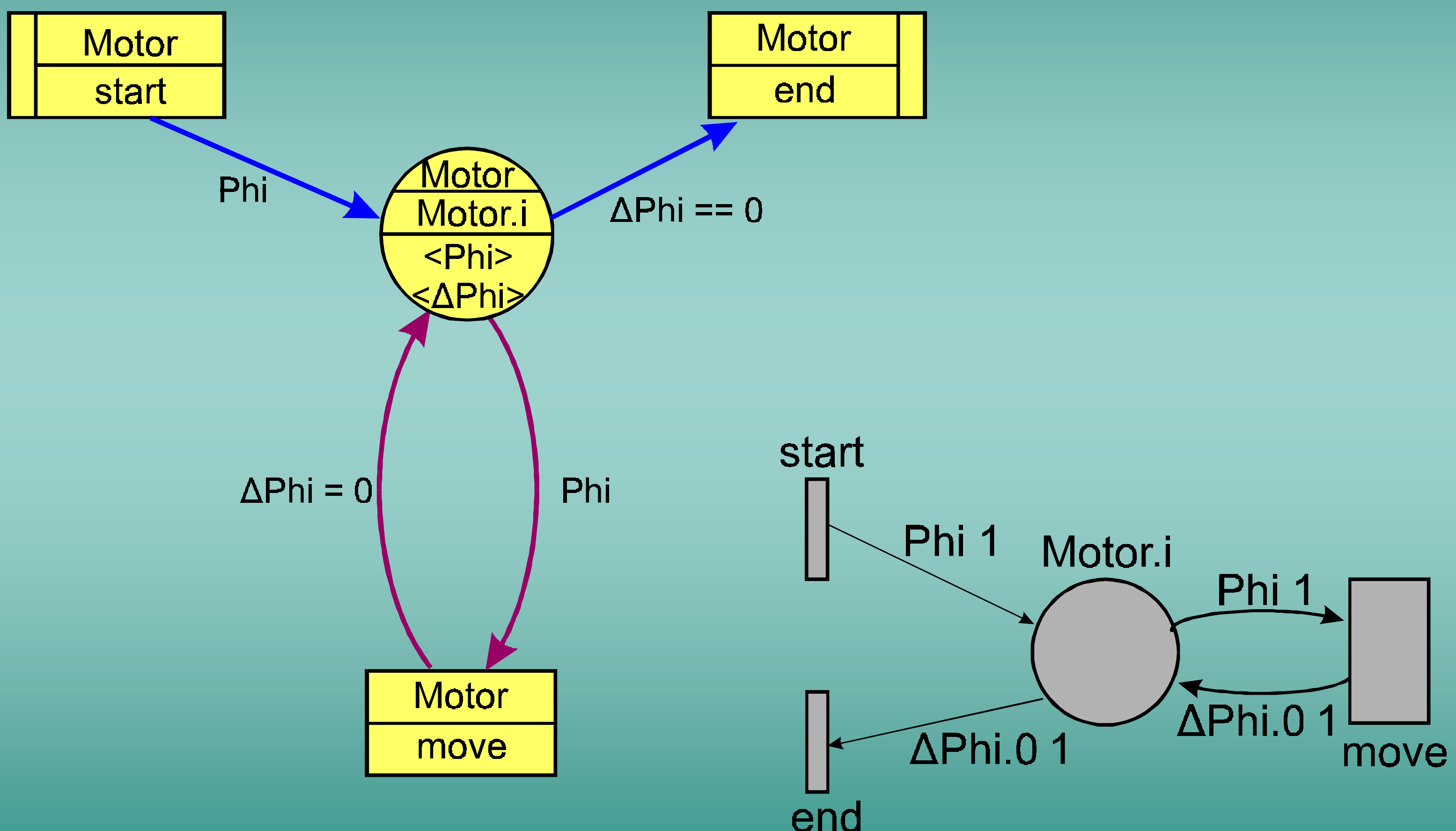


The Refinement of the Subprocess “gotoCoords” of the Class “Project”



X	Y	
Phi1	C1	M1
Phi2	C2	M2

OPM and Corresponding Coloured Petri Net for the Subproblem: "M_move"



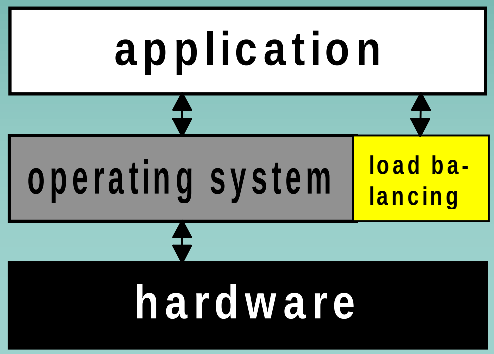
Topics

1. Introduction
2. Object Oriented Design
3. <u>Distribution Procedure</u>
4. Communication Design
5. Hardware Design
6. Analysis Methods
7. Summary

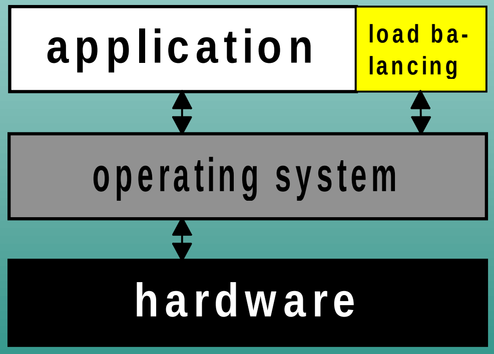
Adaptive Load Balancing: Problems

- load balancing is needed to optimize performance in multiprocessor systems
- load balance highly depends on time
- load balance and its fluctuations are not predictable

Comparison to Other Methods

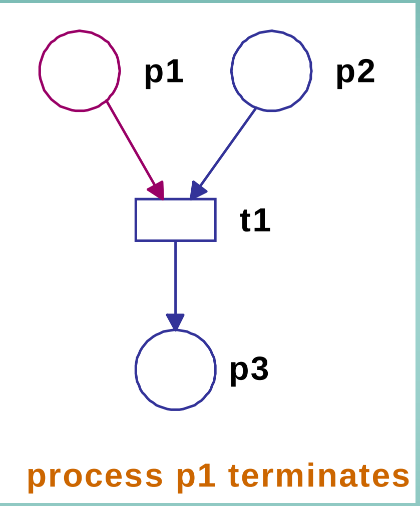


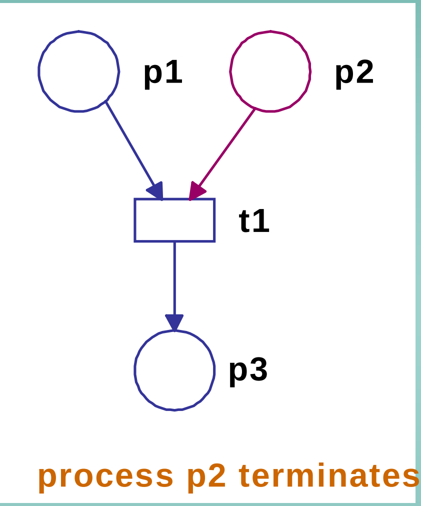
- old:
  central load balancing in the operating system



- new:
  distributed load balancing in the application tasks

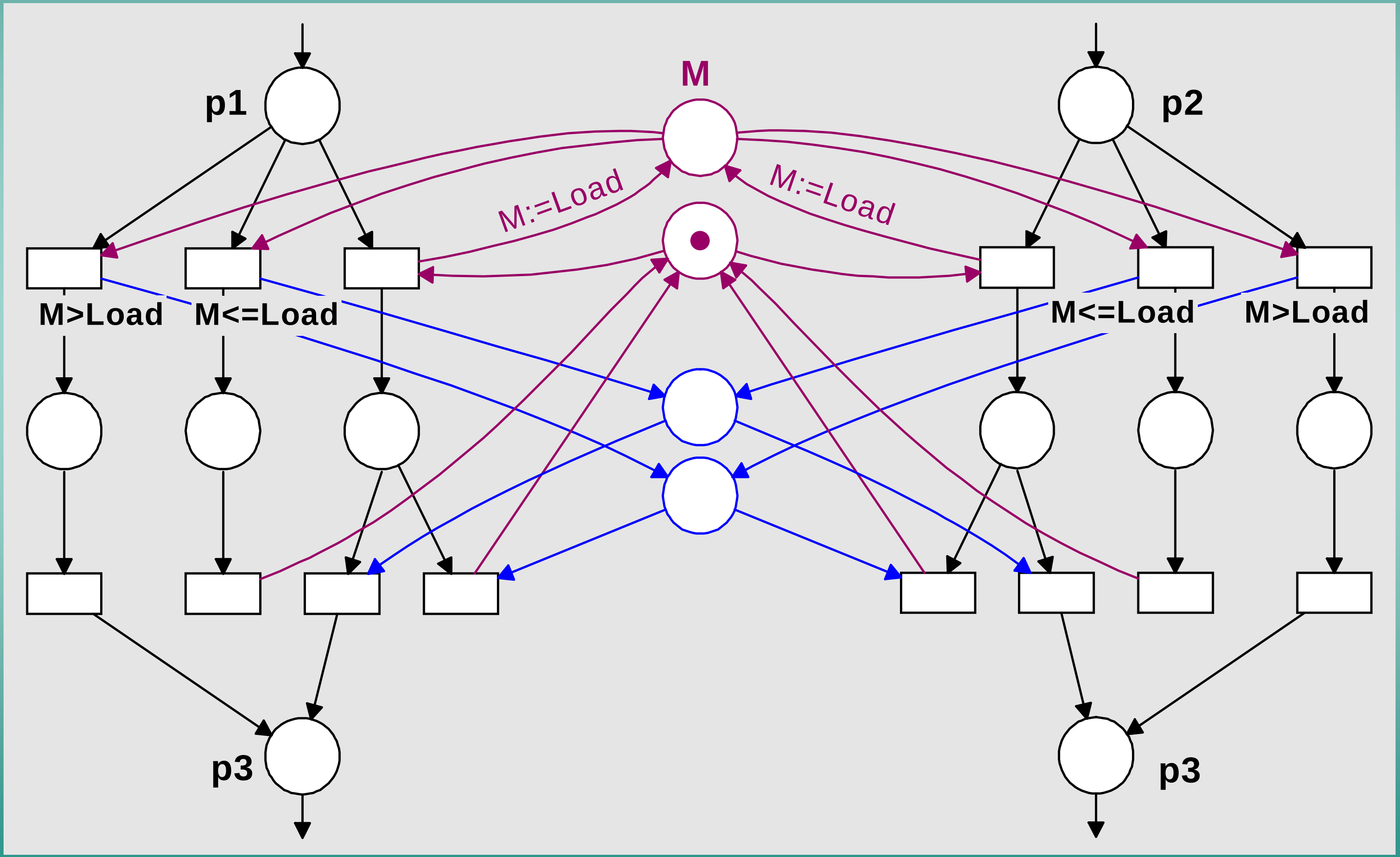
Basic Idea: Flexible Decisions at Synchronization Points



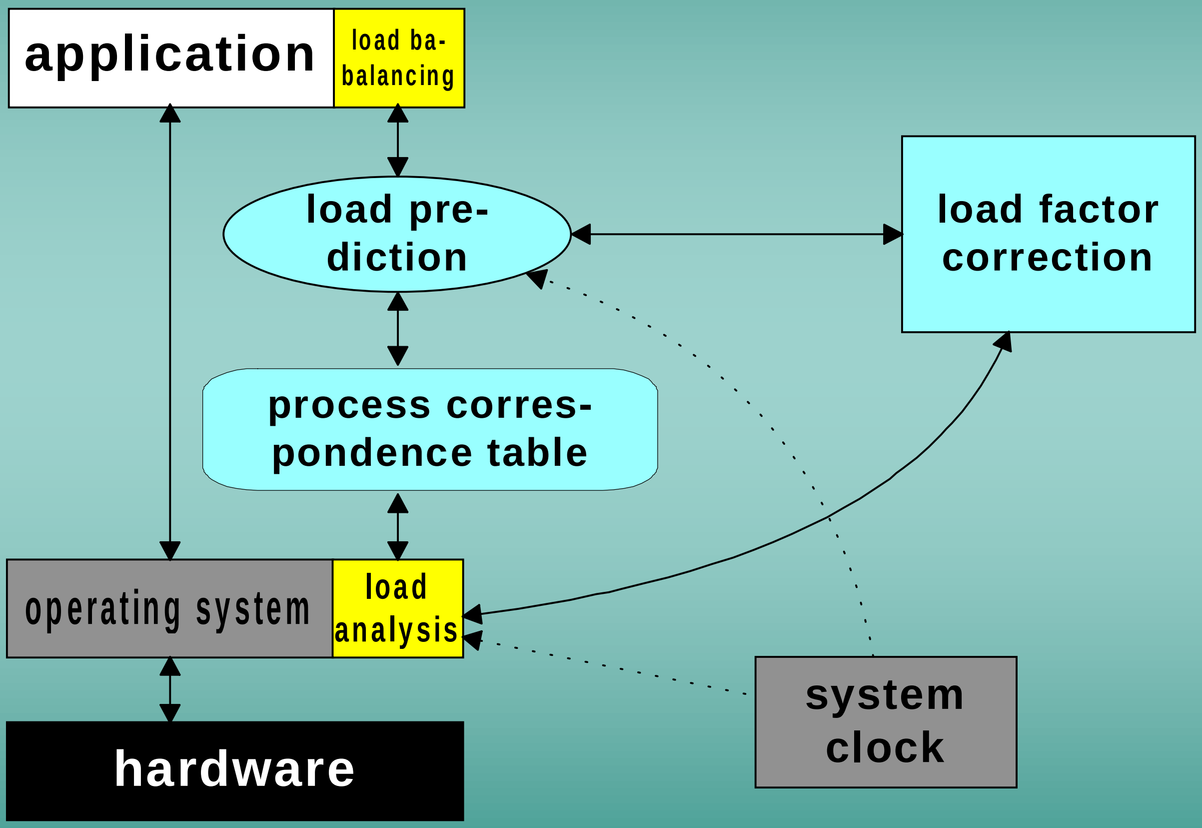


- main rule: the process at the processor with the higher actual load terminates
- may be implemented with the help of Petri Net analysis
- improvement possible by combination with load statistics

Modified Synchronization Point



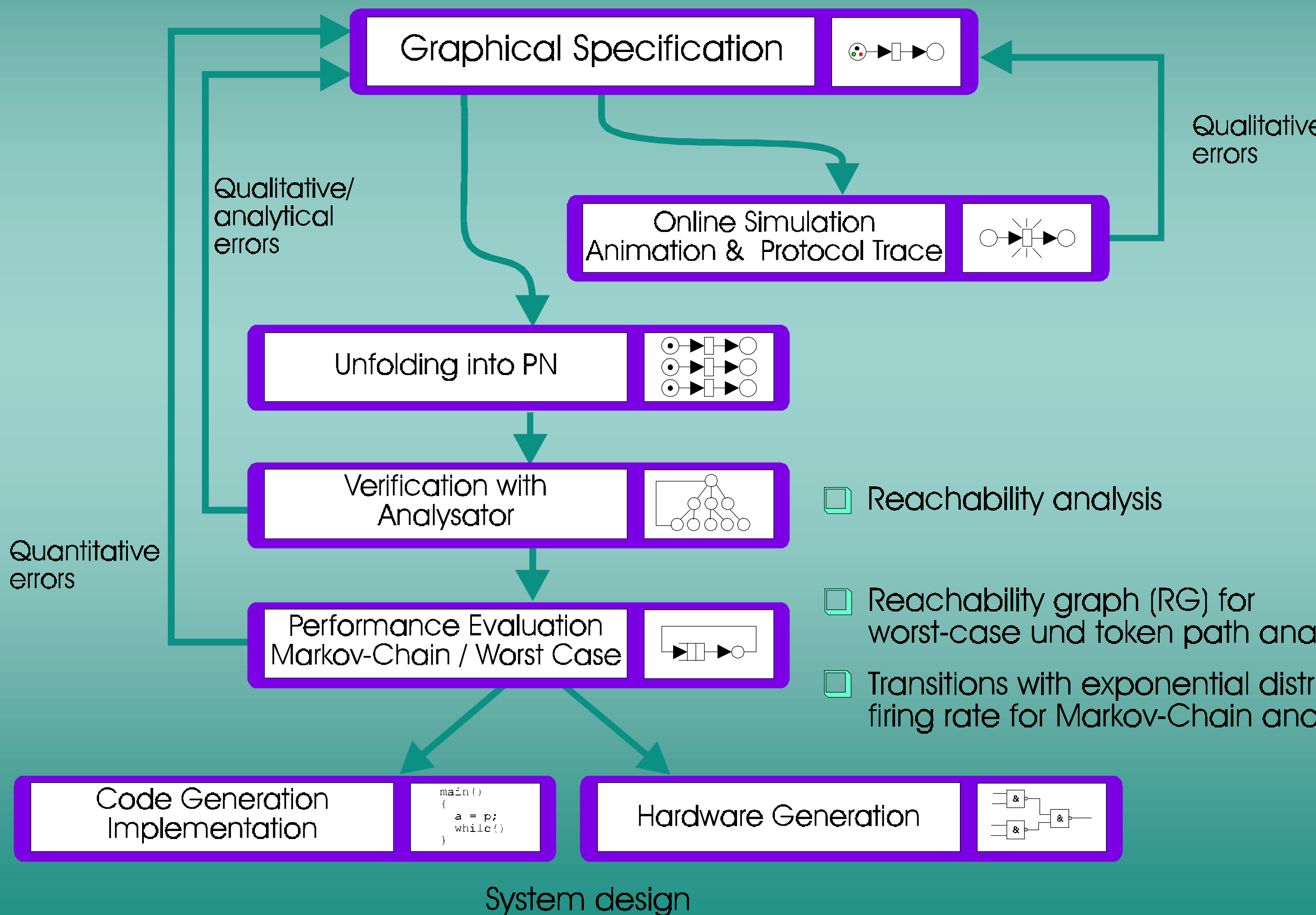
Implemented Structure



Topics

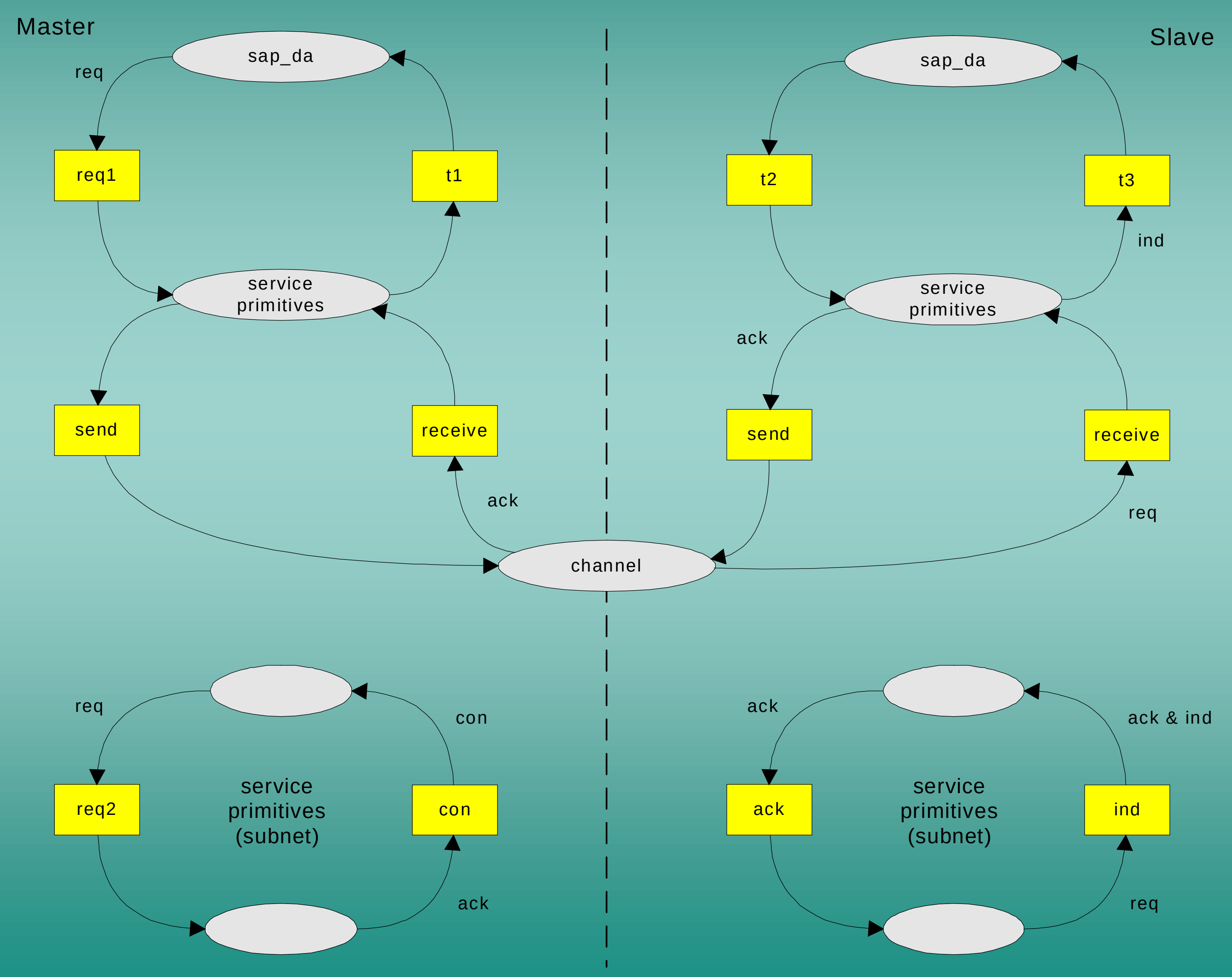
1. Introduction
2. Object Oriented Design
3. Distribution Procedure
4. <u>Communication Design</u>
5. Hardware Design
6. Analysis Methods
7. Summary

Protocol Design with Extended CPNs



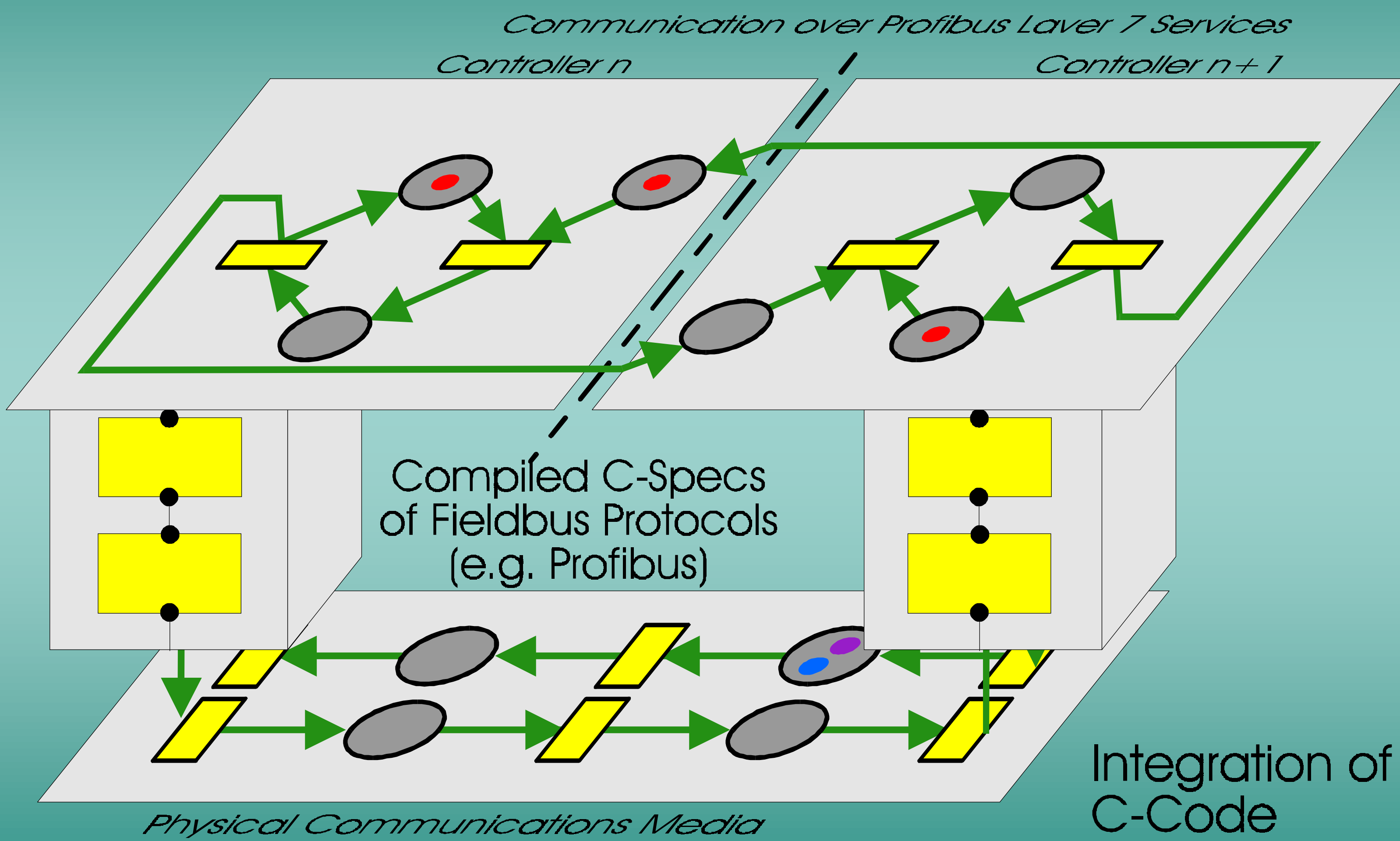
- Reachability analysis
- Reachability graph (RG) for worst-case und token path analysis
- Transitions with exponential distributed firing rate for Markov-Chain analysis

Example for Service Specification



Specification of Distributed Controllers

e.g. with Fieldbus Communication



Integration of C-Code Protocol Specifications

Topics

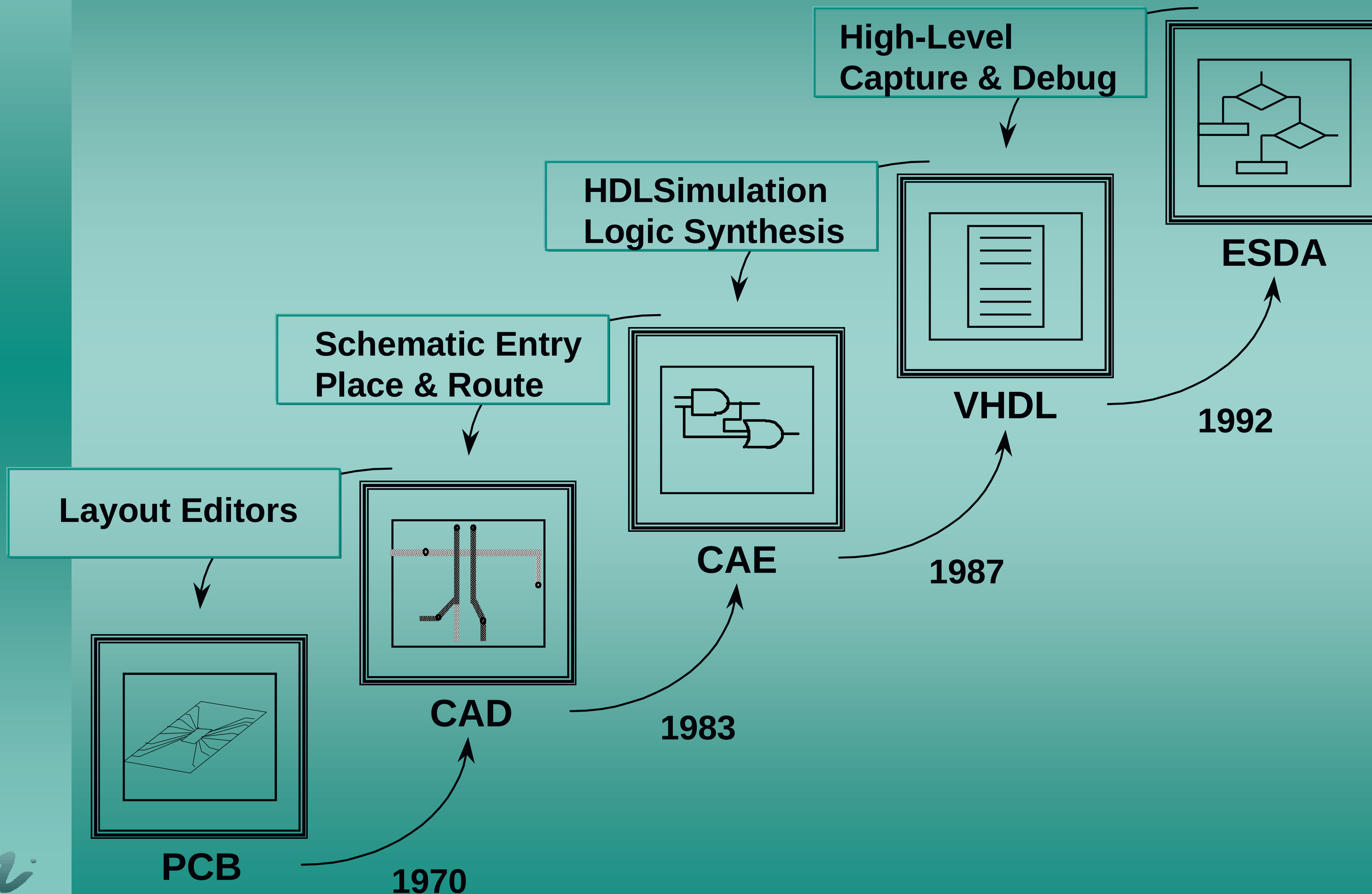
1. Introduction
2. Object Oriented Design
3. Distribution Procedure
4. Communication Design
5. <u>Hardware Design</u>
6. Analysis Methods
7. Summary

5

ASIC-Design - Current Environment

- Modern ASIC:
  - synchronous design with a synchronous reset
  - high complexity (>200k), critical timing (>50MHz)
- High volume of VHDL-code => Use of ESDA
- ESDA: graphic input, the specification can be simulated, error search, redesigns and design hand overs, automatic synthesized VHDL.
  Disadvantage: VHDL-code has “only good” quality

Design Environment - Historical Evolution

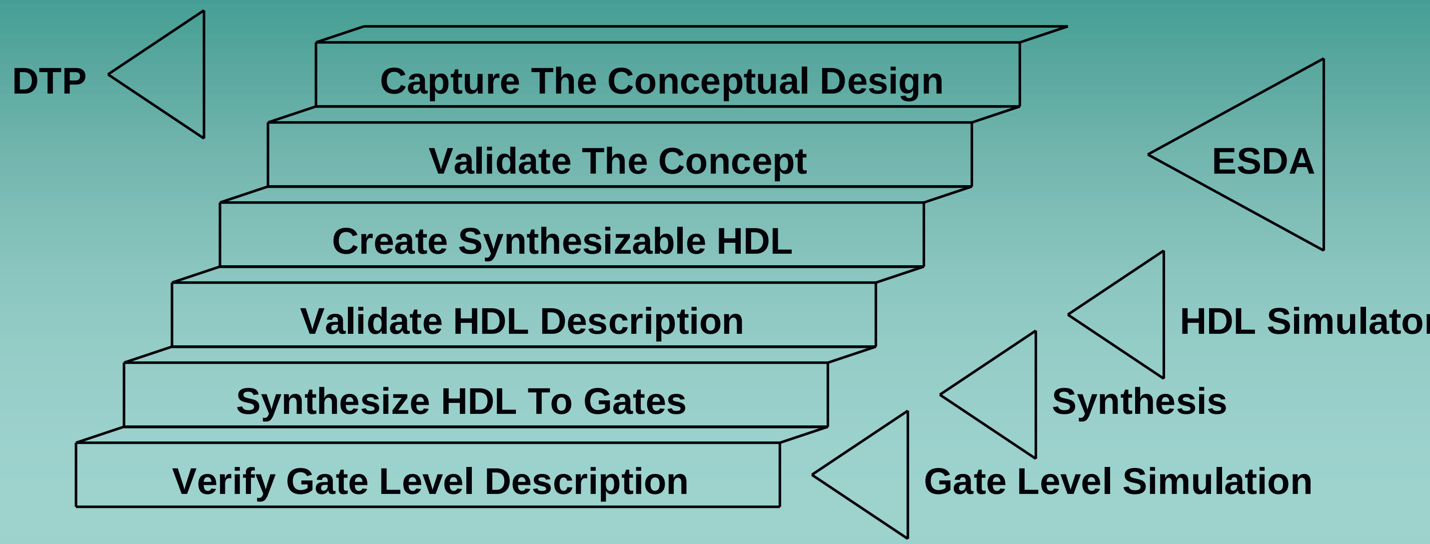


ASIC-Design by Petri Nets

- Petri Nets:
  - the same input possibilities as ESDA, simulation of parallel algorithms on graphical level
  - high level (hierarchical, coloured etc.) Petri Nets
  - statistical analysis of design
  - for synchronous produced signals: representation of the signals as place elements
    - signal conflict values
    - signal values by a dead marking
    - reachable signal values etc

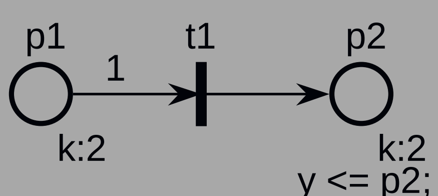
5

Design Steps



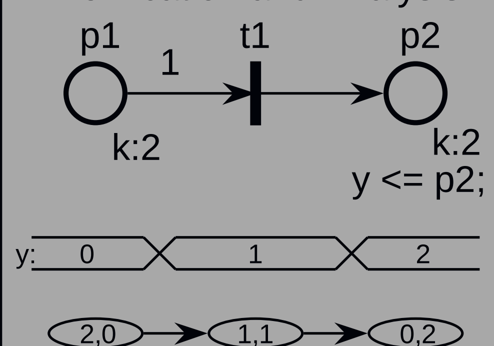
Capture The Concept. Design

Petri Net Design



Validate The Concept

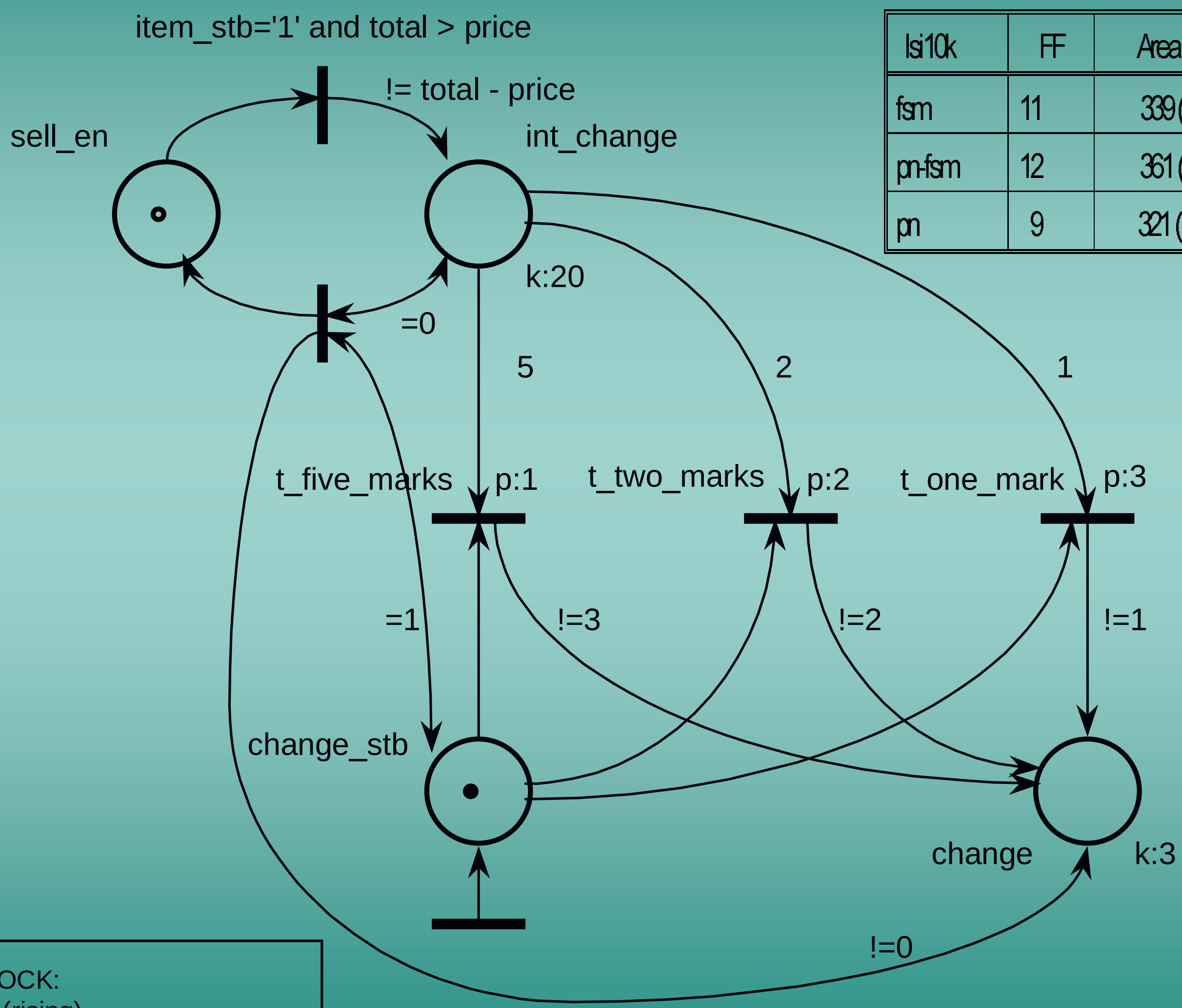
Verification and Analysis



Create Synthesizable HDL

```
architecture pnsd of ...
begin
  p1_t1 <= (p1 >= 1);
  t1_p2 <= (p2 <= 1);

t1 <= p1_t1 and t1_p2;
```

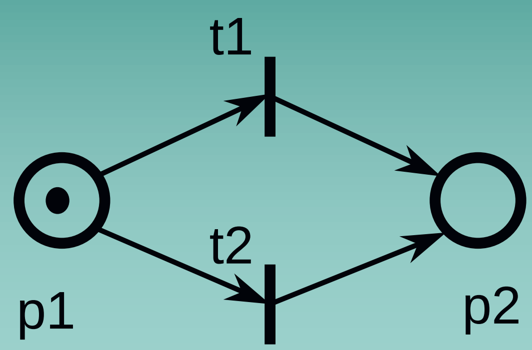
From FSM to PN



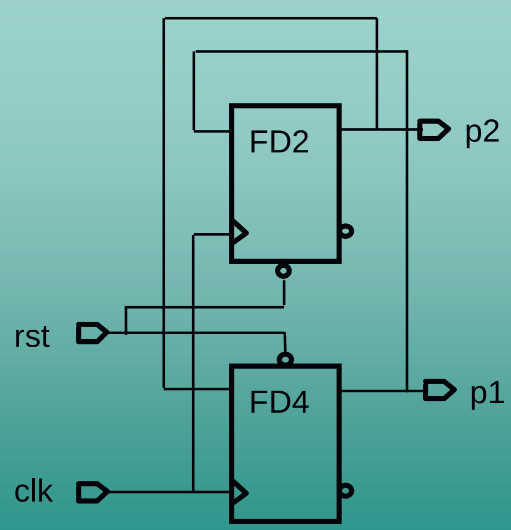
lsi10k	FF	Area (Tmin)	Area (Tmin) FSM
fsm	11	339 (7.4ns)	350 (7.2ns)
pn-fsm	12	361 (7.6ns)	316 (6.9ns)
pn	9	321 (10.4ns)	458 (6.1ns)

CLOCK:
clk (rising)
RESET:
rst (low active, synchronous)

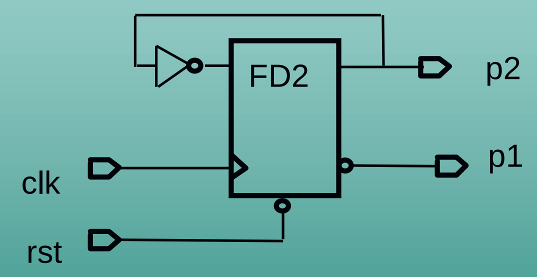
Logic Synthesis



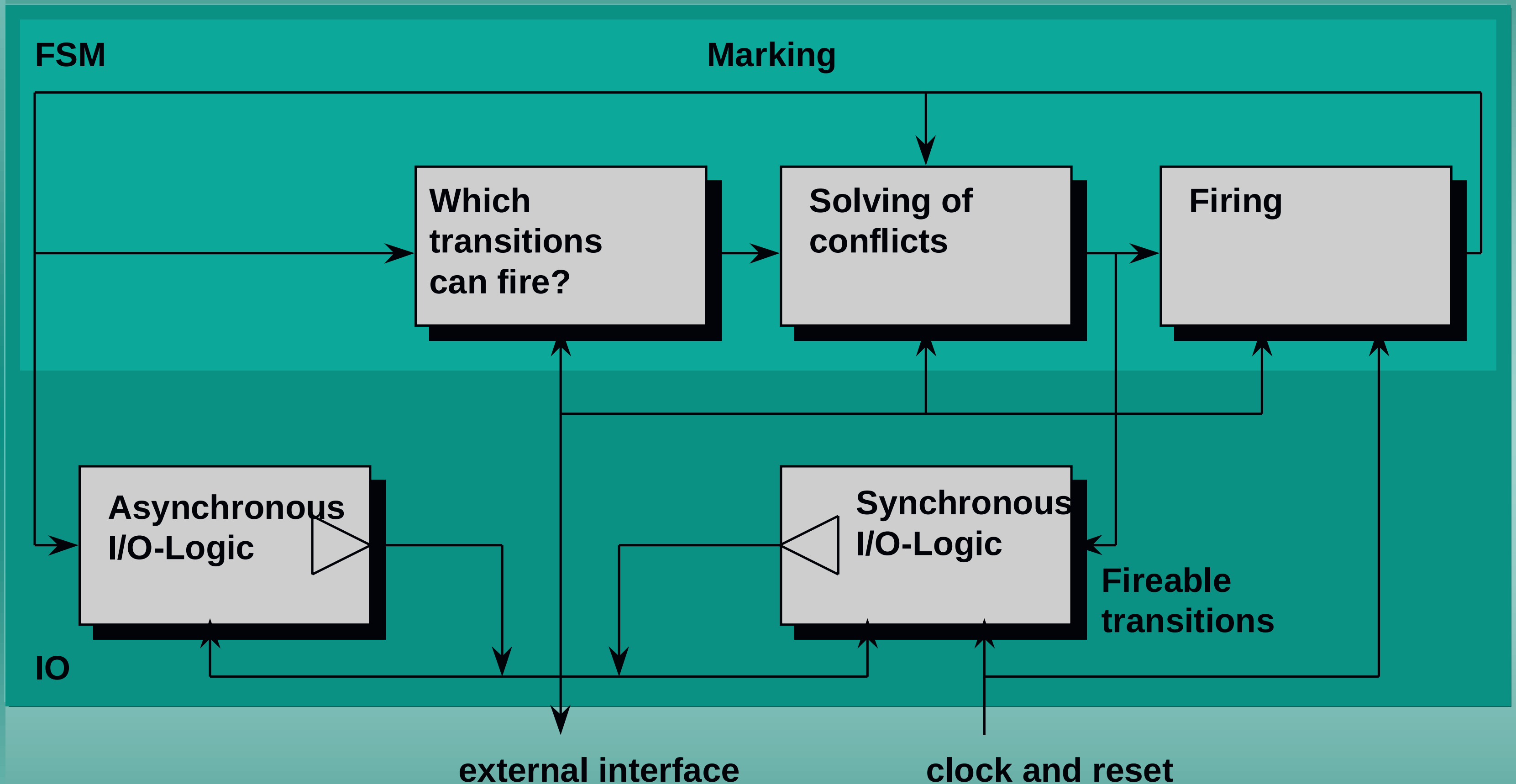

```
t1 <= (p1 = 1) and (p2 = 0);
t2 <= (p2 = 1) and (p1 = 0);

process (clk,rst)
begin
  if rst = `0' then
    p1 <= 1; p2 <= 0;
  elsif clk'event and clk = `1' then
    p1 <= p1 - getm(t1,1) +
getm(t2,1);
    p2 <= p2 - getm(t2,1) +
getm(t1,1);
  end if;
end process;
```



(p1,p2)=
{(1,0),(0,1)}

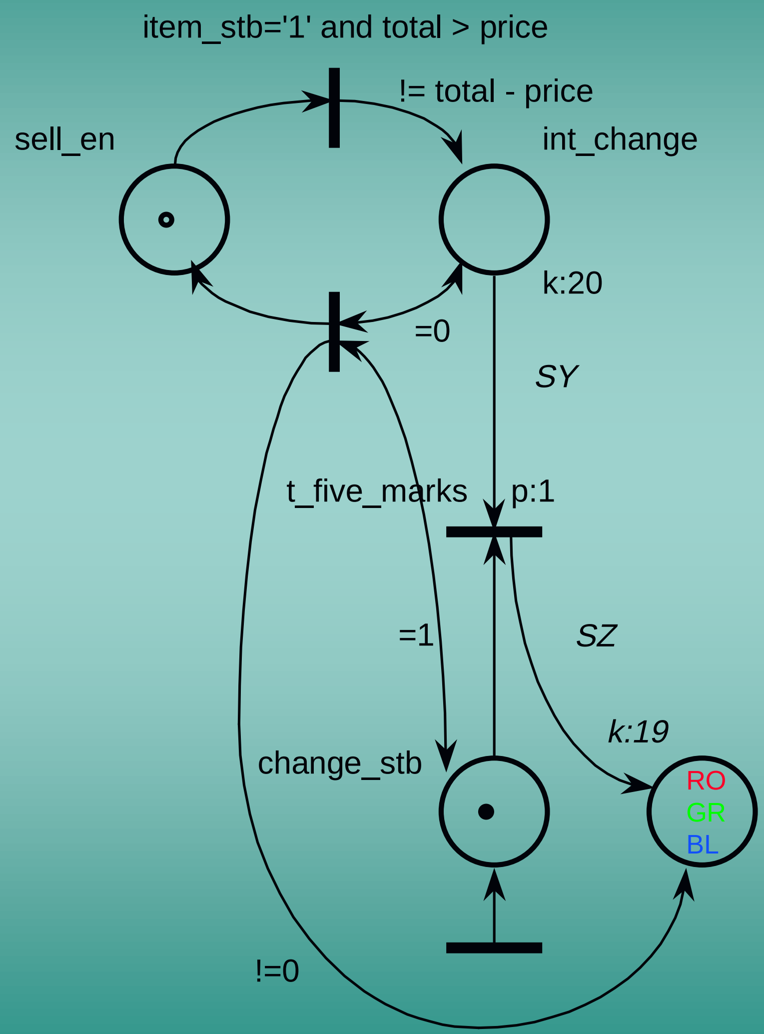


Export to VHDL Description



- Synthesizable VHDL Code (IEEE-1076)
- Hierarchical design for optimization of FSM

From PN to Coloured PN



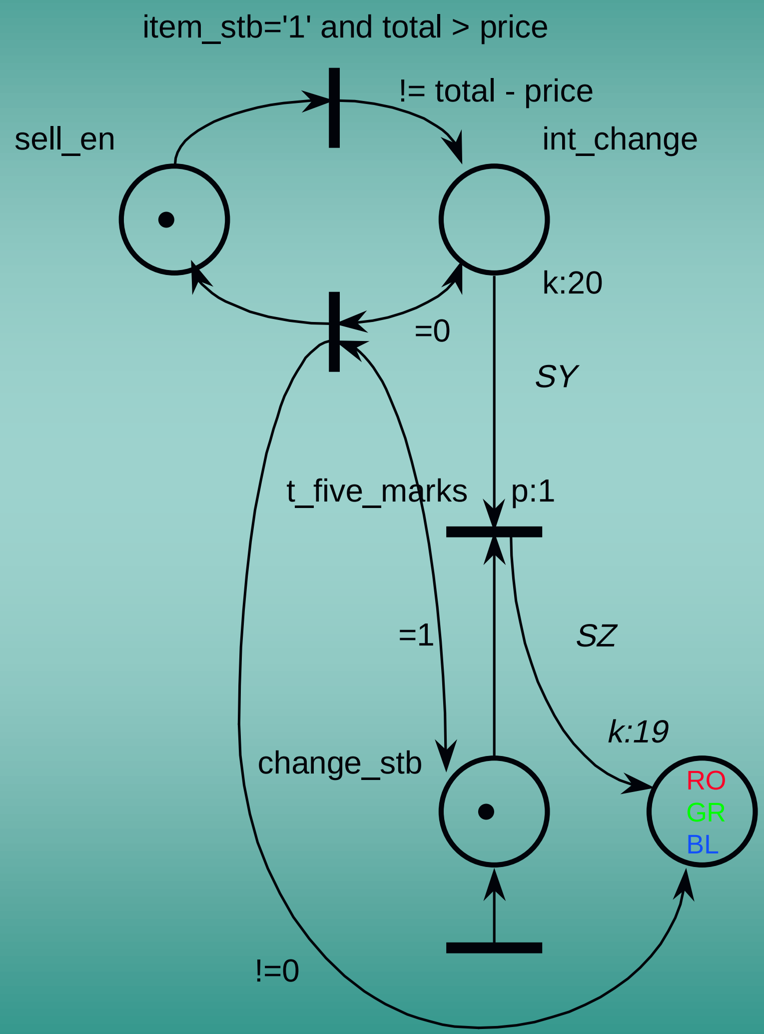
COLOURS CONVERSION

SY	SZ
5 SW	1 RO
2 SW	1 GR
1 SW	1 BL

PRIORITIES

COL.	PRIOR.
RO	1
GR	2
BL	3

From PN to Fuzzy Evaluated CPN



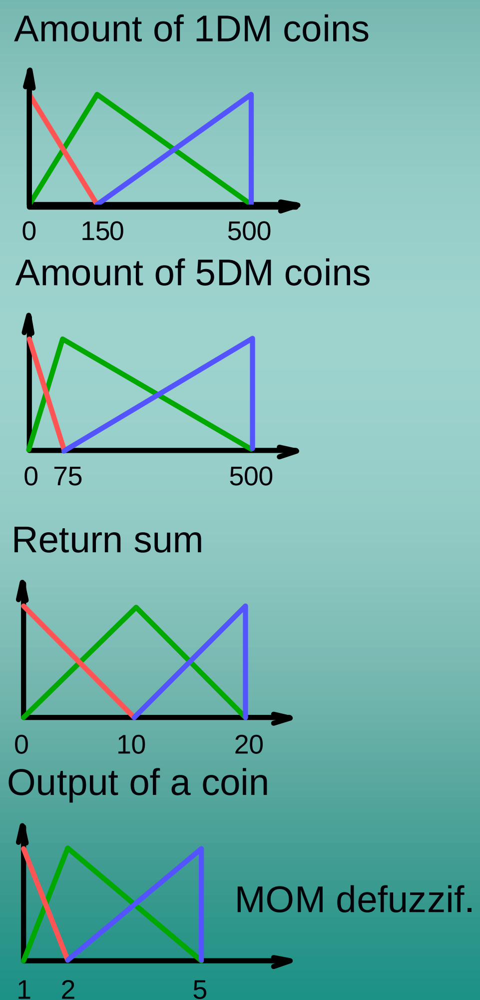
COLOURS CONVERSION

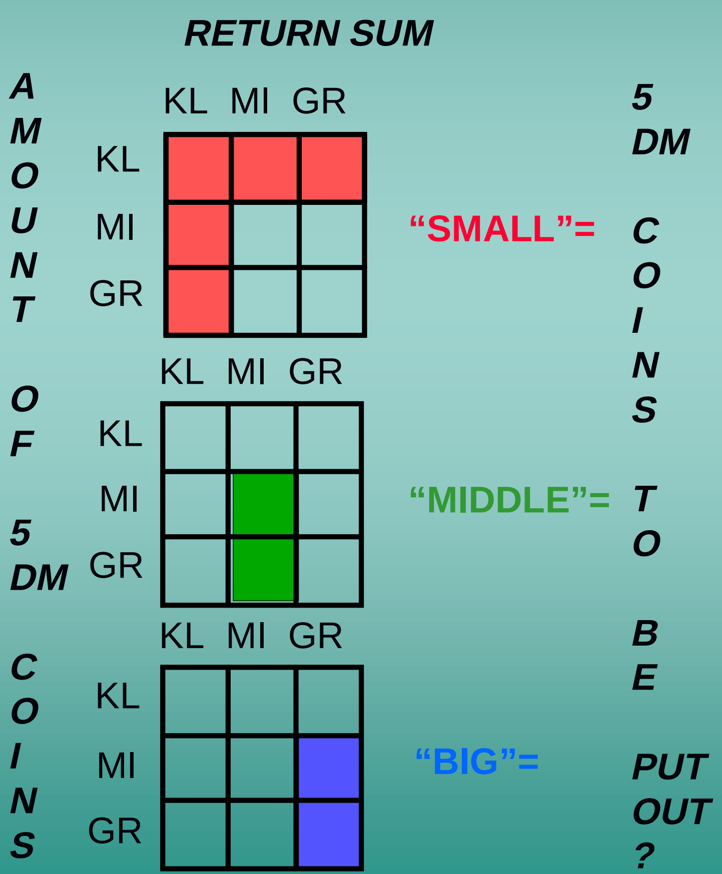
SY	SZ
5 SW	1 RO
2 SW	1 GR
1 SW	1 BL

FUZZY RULES:

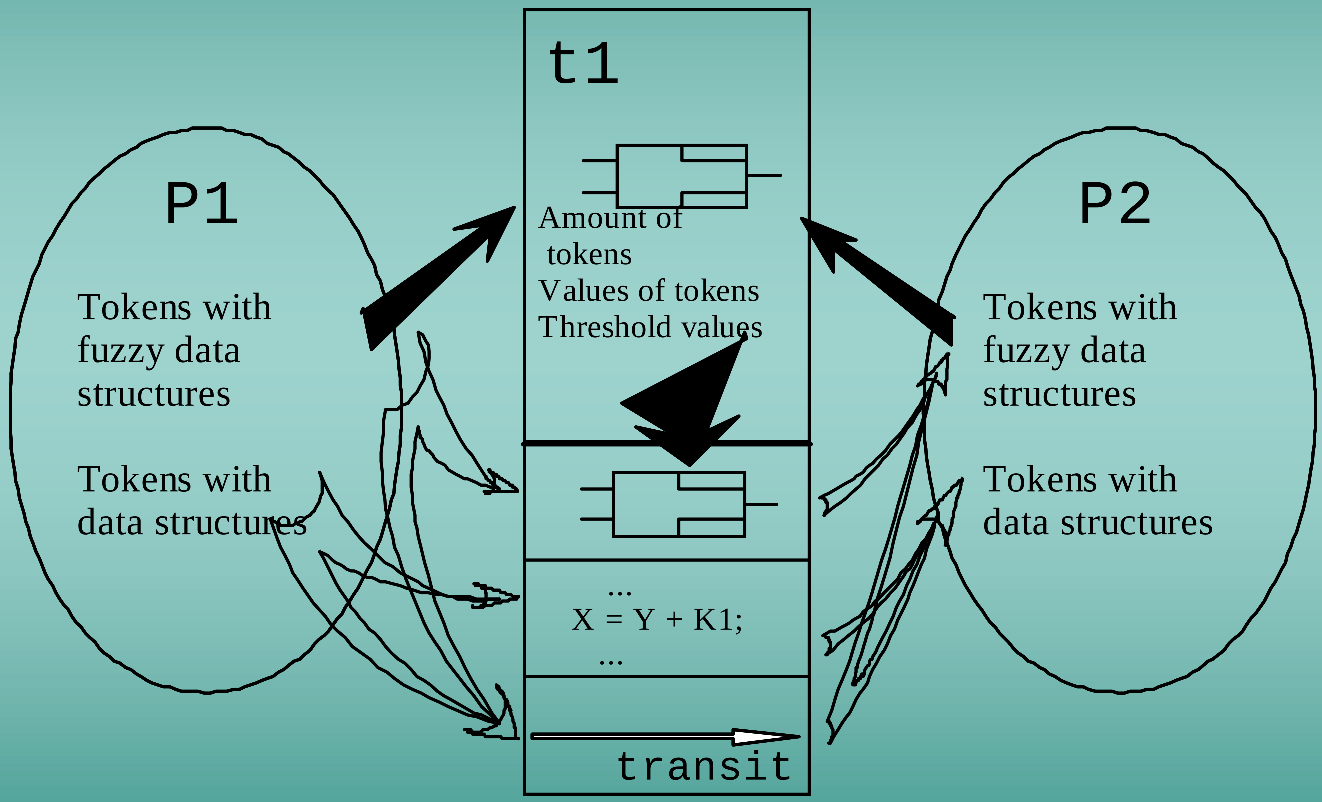
1. The bigger a sum, the bigger coins should be put out;
2. The smaller the amount of big coins, the smaller the relation “big/small coins” in an output;
3. The smaller the amount of 1DM coins, the more seldom they will be put out.
Attention: The return sum saved in the place int_change is varying during the output

Membership Functions and Variable Links



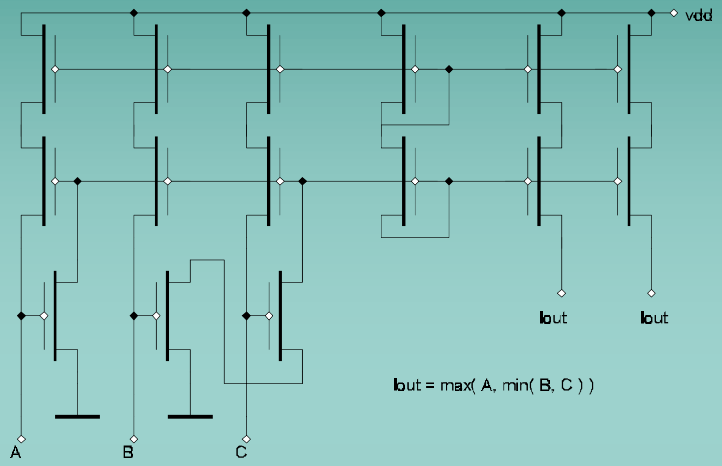


Fuzzy Evaluated Transition



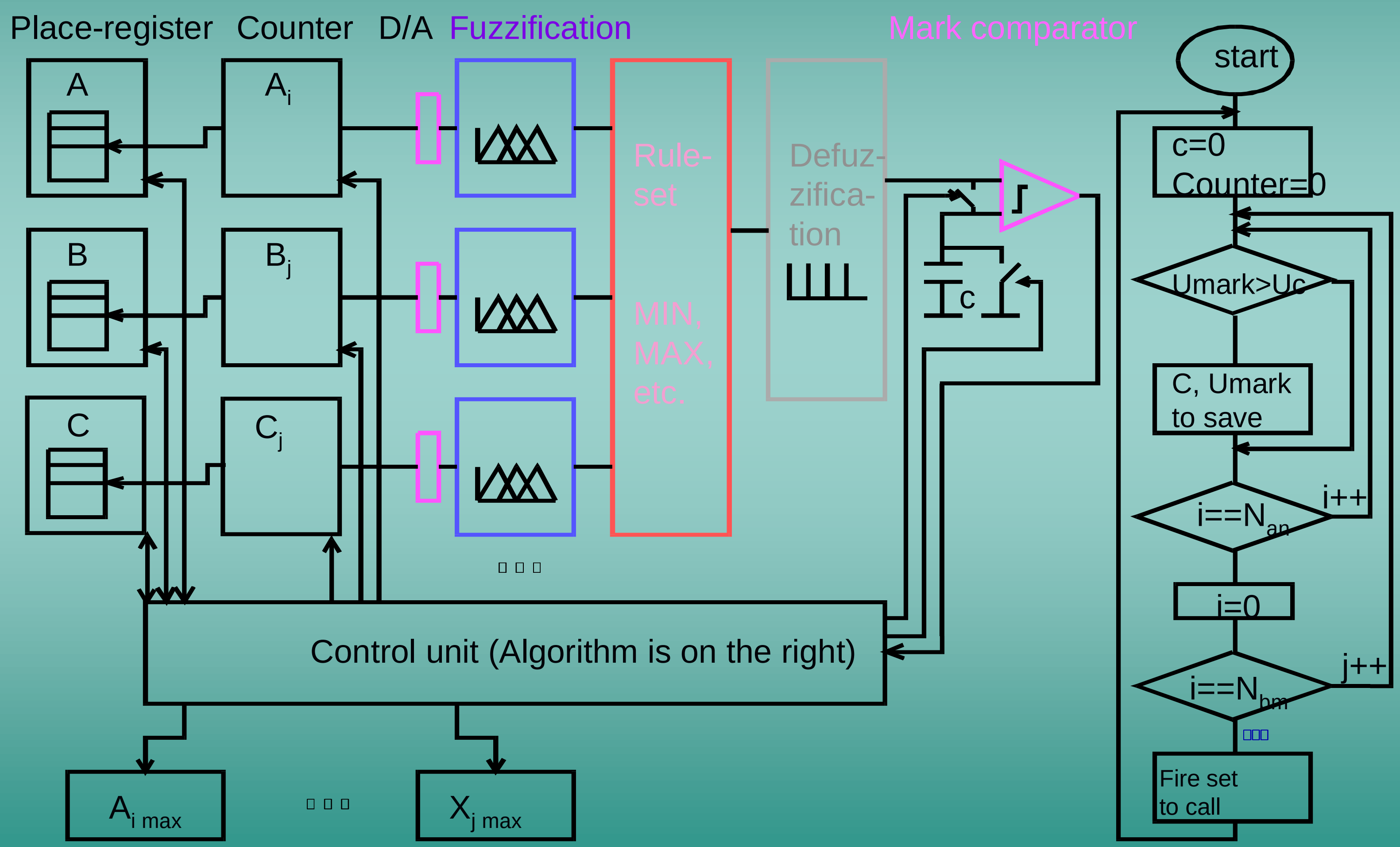
Why Analog-Digital Realization?

MAX-MIN complex operator:



Realization: Goodness of. ...	digital	analog-digital	Software
Performance	+	++	--
Hardware expenses	-	+	-
Development expenses	+	-	++
Precision	+	-	++

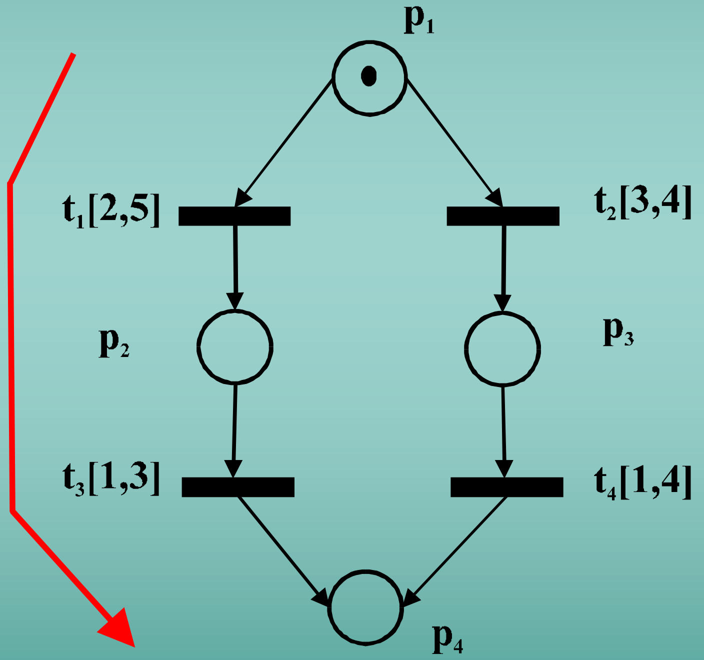
Realization Conception of the Fuzzy Evaluation Function

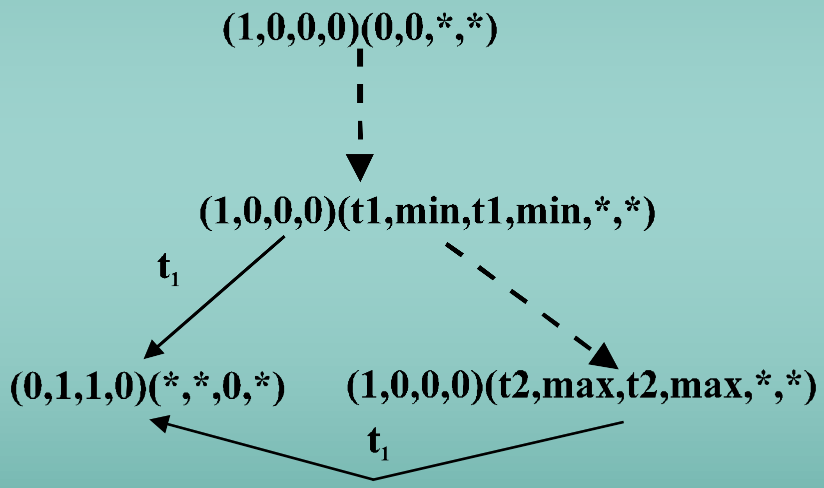


Topics

1. Introduction
2. Object Oriented Design
3. Distribution Procedure
4. Communication Design
5. Hardware Design
6. <u>Analysis Methods</u>
7. Summary

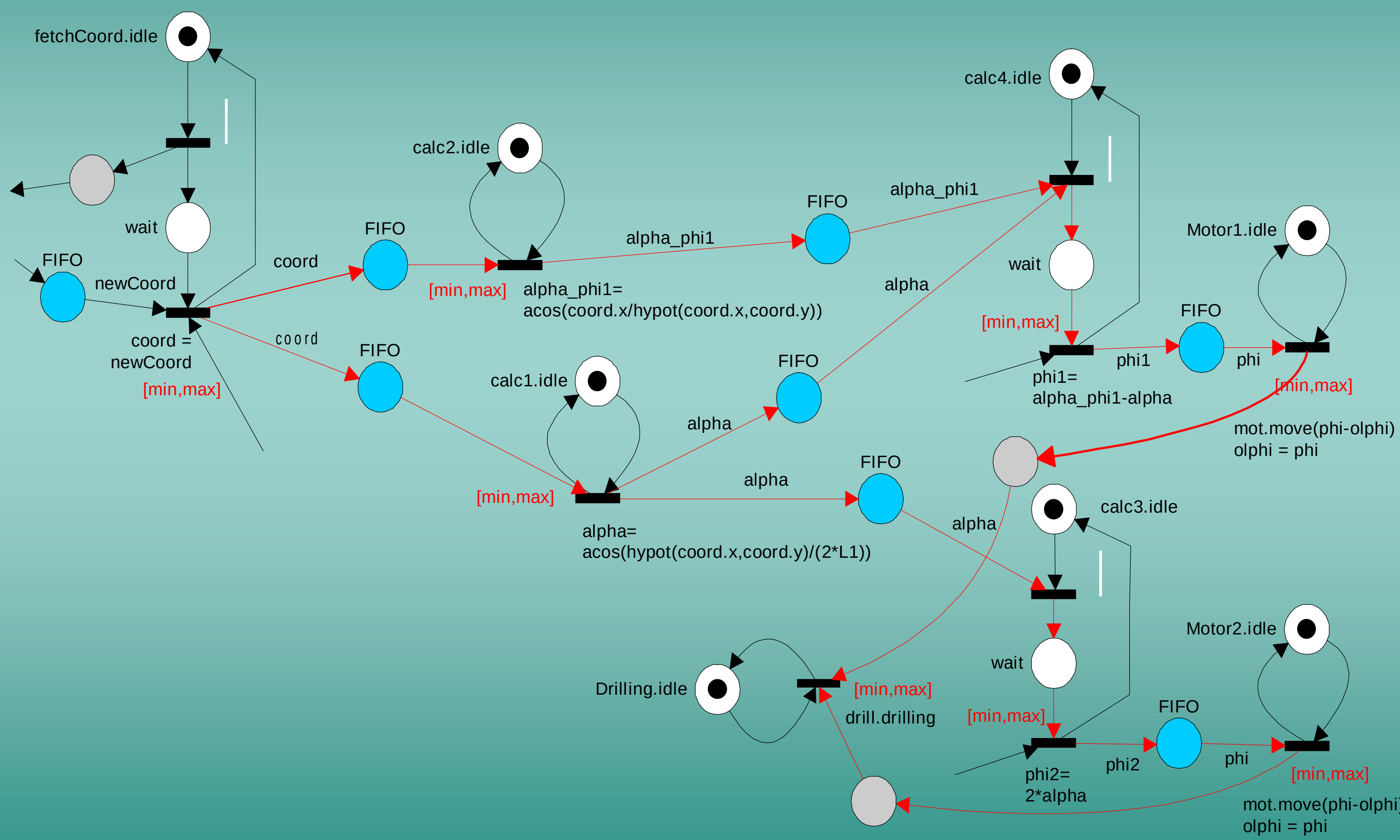
Min-/Max-Times





Tmax=Min{t1,max,t2,max}+t3,max

Min-/Max-Times: Example



Petri Net

PN = (P, T, F, v, k, m)

↓

State Net

ZN = (Z, E, G(E))

- Reliability investigation only by reachability graph
- Classical reliability methods useable

PN = (P, T, F, v, k, m) → ZN = (Z, E, G(E))

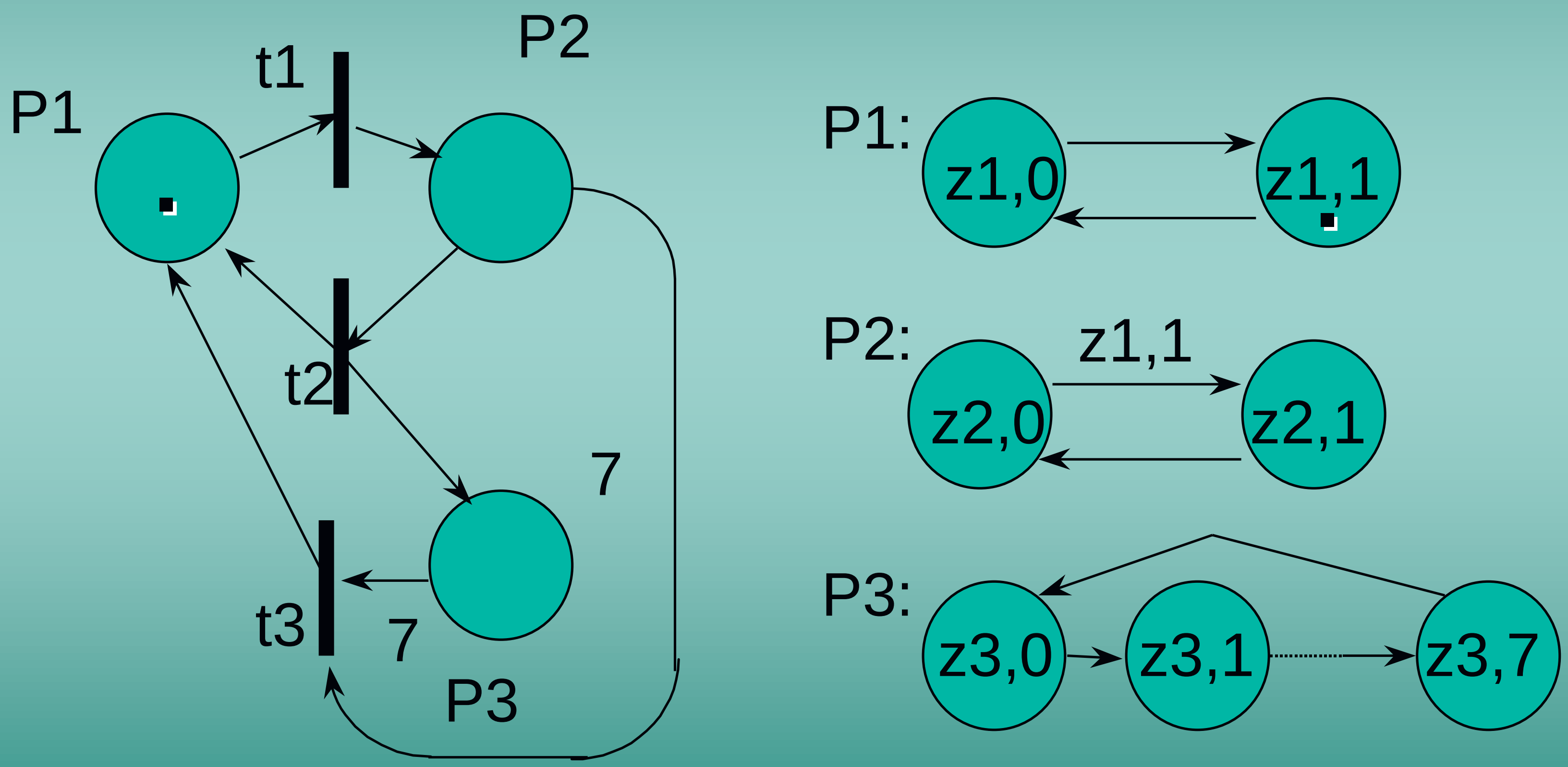
Basic Idea

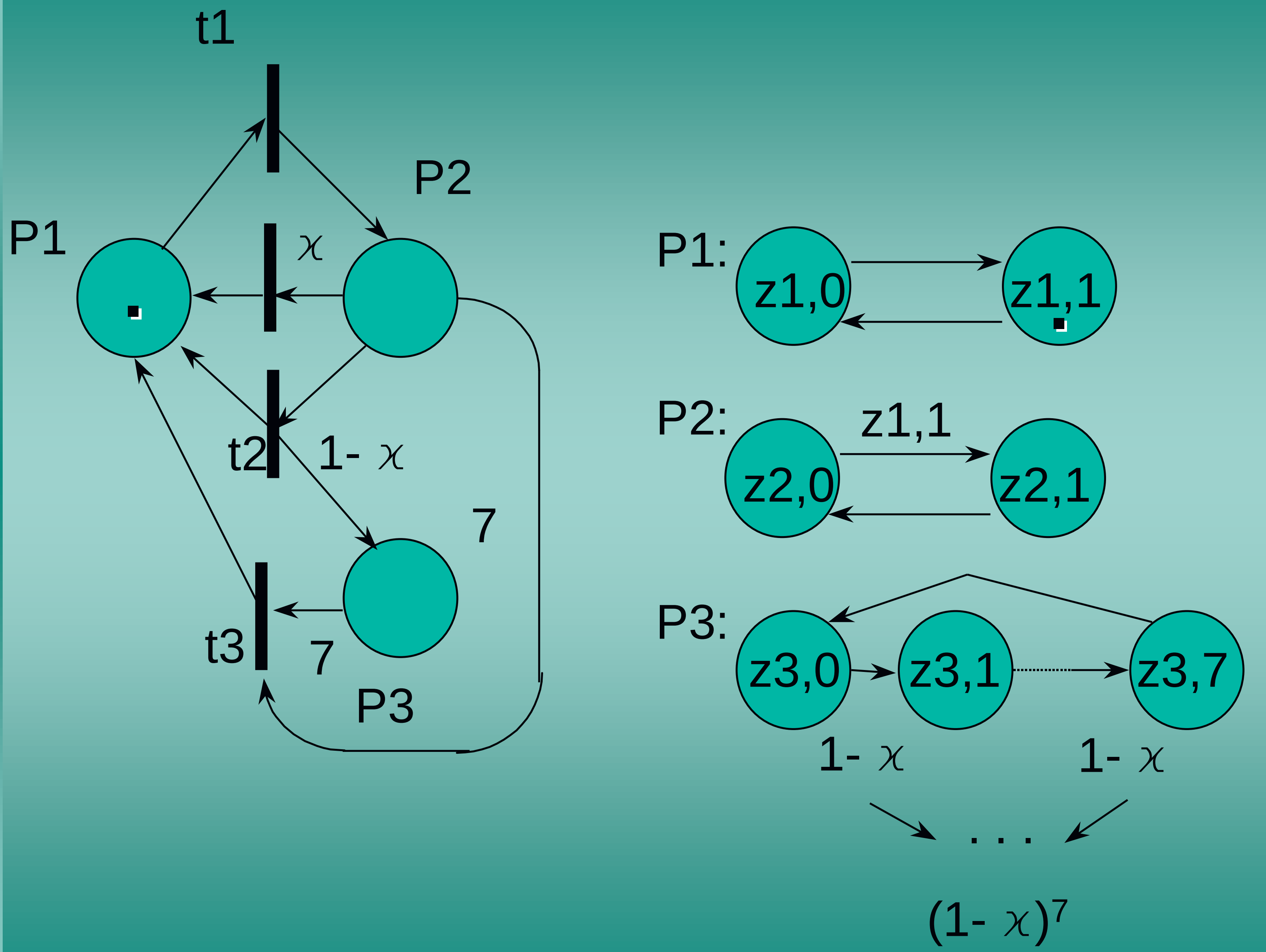
| Zi | = K(pi) + 1   i=1(1)n

Z = z1 & z2 & ... & zn

G(E) = {g(e1), g(e2), ..., g(en)}

Example




t1
P2
P1
x
t2
1- x
7
t3
7
P3
P1:
z1,0
z1,1
P2:
z1,1
z2,0
z2,1
P3:
z3,0
z3,1
z3,7
1- x
1- x
. . .
(1- x)7

Topics

1. Introduction
2. Object Oriented Design
3. Distribution Procedure
4. Communication Design
5. Hardware Design
6. Analysis Methods
7. <u>Summary</u>

Summary

- With the object oriented design method complex facts can be fixed concisely and comprehensively and refined hierarchically.
- The mapping of processes onto processor elements can be optimized by means of the formal notation.
- The required communication structures can be specified, simulated and implemented with formal methods.
- The design methodology allows the generation of hardware components based on the given description technique.
- By means of formal analysis the correctness of the design can be verified comprehensively.