

Teaching Principles of Petri Nets in Hardware Courses and Student's Projects

KUBÁTOVÁ Hana
kubatova@fel.cvut.cz

Outline

- Teaching context
- Course content
- Experiments - student's projects
- Conclusions

Teaching context

- CTU - the oldest technical university in central Europe (founded in 1707), ... 20 000 students
- FEE - its greatest faculty (... 5 000 students)
- DCSE - the greatest dept. in FEE
 - 350 bachelor students every year
 - 200 magister students every year
 - 160 magisters (Ing.) every year

Teaching context

.... but everybody would like to
study software
(or computer networks)

5 magister specializations

- Software engineering
- Computer networks
- System programming
- Computer graphics
- Hardware design

Teaching context *for me*:

- ❑ Petri Net formalism in hardware design course - “Architecture of peripheral devices”
- ❑ student individual or group projects
- ❑ from Place/Transition nets to Coloured Petri nets
- ❑ direct hardware implementation (synthesized VHDL for FPGA)

Teaching context

- ❑ attractiveness - self-explanation, clear and practical examples
- ❑ to be as simple as possible
- ❑ if hardware ... then the modern design methods and kits
- ❑ if labs then their final product must "do" something - not only simulation!
- ❑ if formalism then only where it is necessary

Why Petri Nets?

- ❑ mathematically defined formal model
- ❑ ease of understanding, declarative, logic based and modular modeling principles, graphically represented
- ❑ able to model asynchronous and parallel actions without any coupling to internal clock frequency

Seminars

- to present essential features of Petri nets: the principles of **duality**, **locality**, **concurrency**, **graphical** and **algebraic representation**
- using clear example:

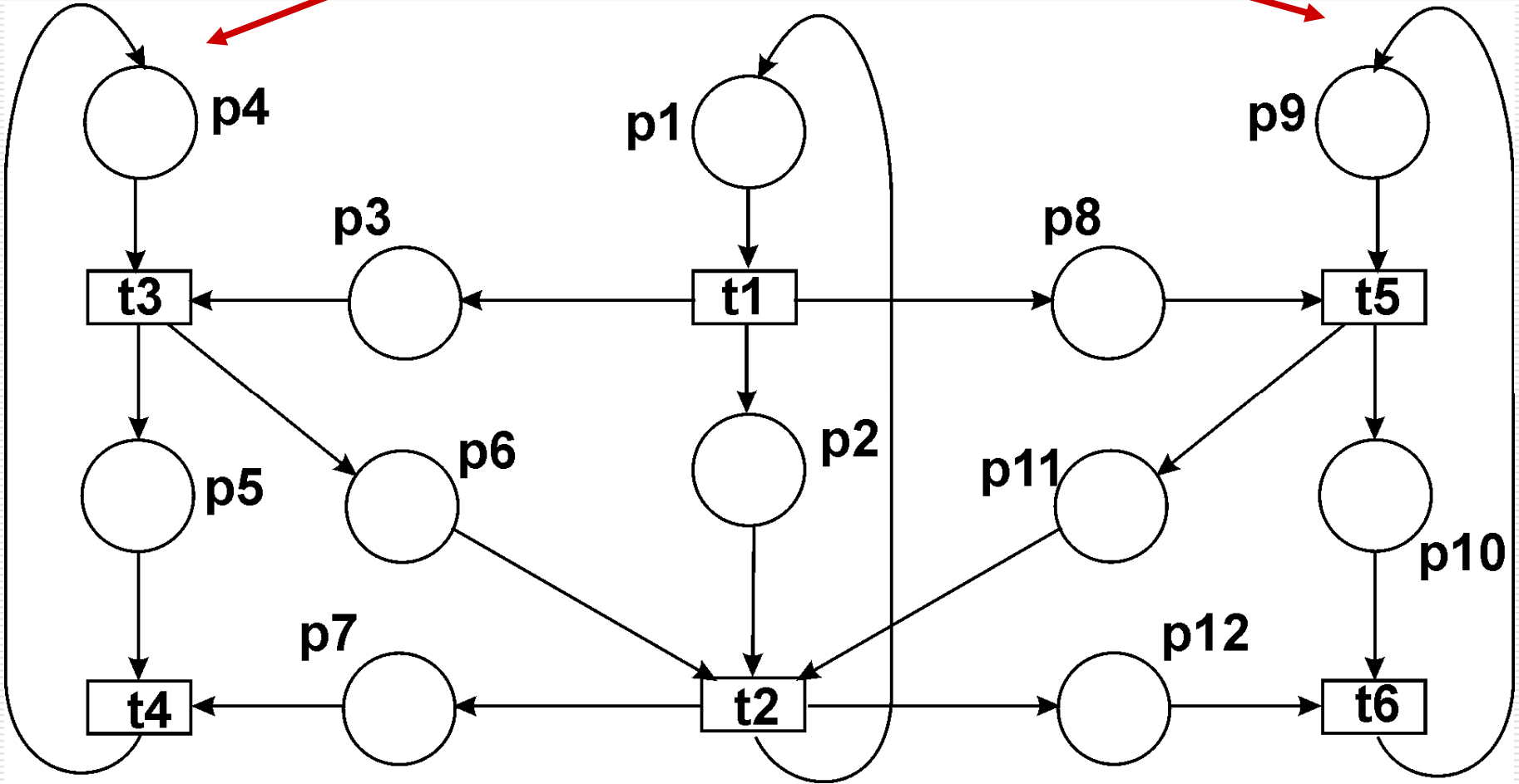
Example

- printers communicating with a control unit that transmits data according to the *handshake* scheme
- control unit - STROBE ... "data valid"
printers - ACK ... "data are printing"
to falling edge of a STROBE signal ...
all printers must react by the falling edges of ACK signals to obtain the next portion of data

■ ■ ■ ■

-
- the aim is not a lecture given by me but to construct their own PN model

1 control unit, 2 printers



List of conditions:

- p1: control unit C has a byte prepared for printing**
- p2: control unit C is waiting for signals ACK**
- p3: control unit C is sending a byte and a STROBE signal to printer A**
- p4: printer A is ready to print**
- p5: printer A is printing a byte**
- p6: printer A sends ACK signal**
- p7: control unit C sends STROBE = 0 to A**
- p8: control unit C is sending a byte and a STROBE signal to printer B**
- p9: printer B is ready to print**
- p10: printer B is printing a byte**
- p11: printer A sends ACK signal**
- p12: control unit C sends STROBE = 0 to B**

List of actions:

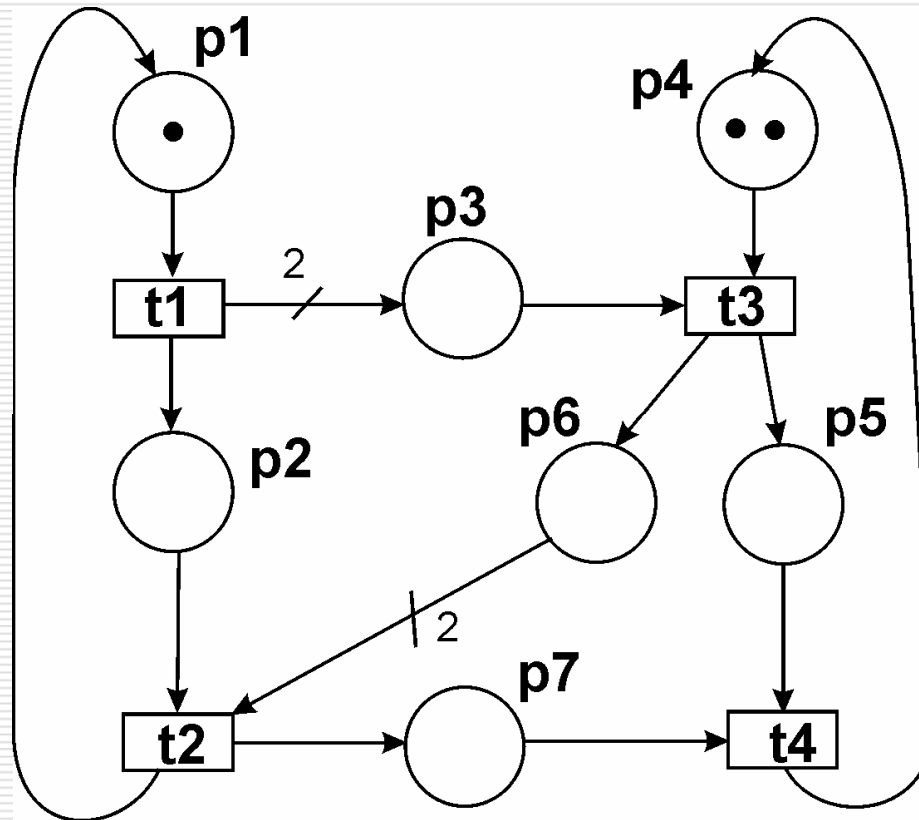
- t1: control unit C sends STROBE = 1**
- t2: control unit C sends STROBE = 0**
- t3: printer A sends ACK = 1**
- t4: printer A sends ACK = 0**
- t5: printer B sends ACK = 1**
- t1: printer B sends ACK = 0**

Separating or identifying passive elements (conditions) from active elements (actions) is a very important step in the design of systems. This duality is strongly supported by Petri nets.

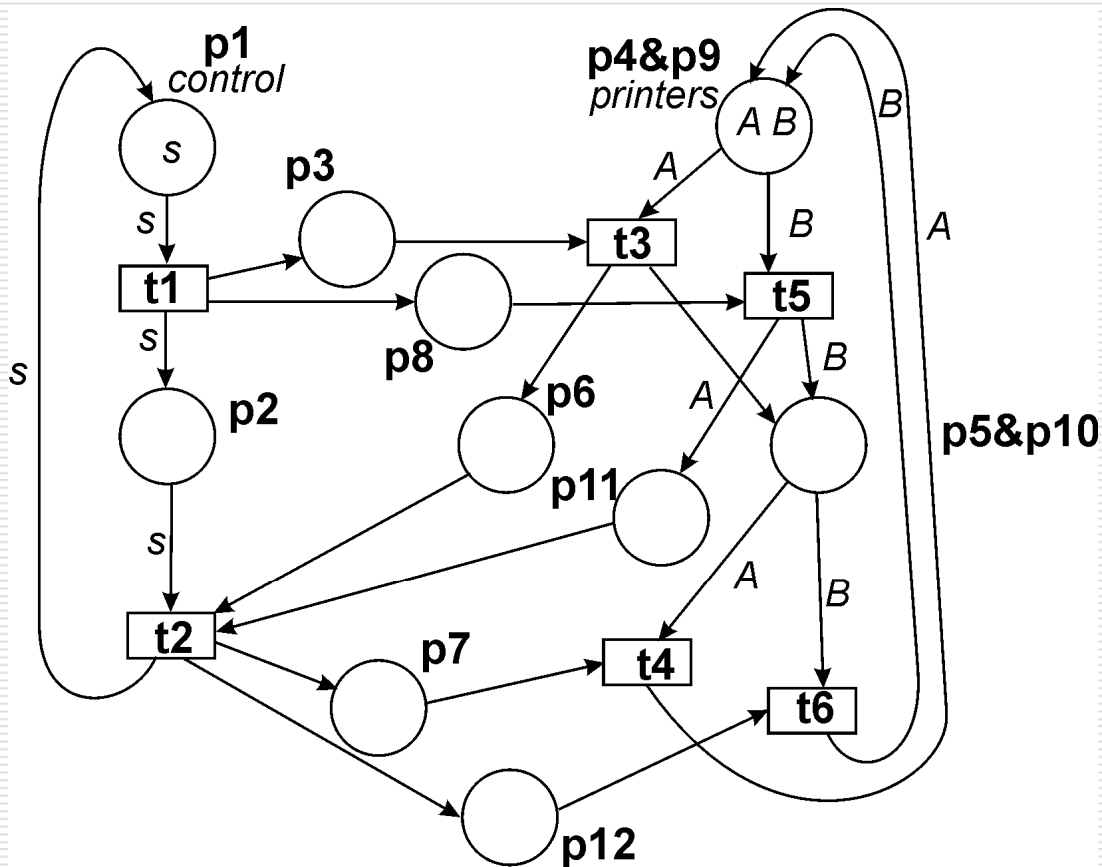
teaching principles of:

- ❑ **duality** - passive elements are represented by places and active elements, are represented by transitions
- ❑ **locality** - the behavior of a transition depends exclusively on its locality
- ❑ **concurrency** - transitions having a disjoint locality occur independently
- ❑ **algebraic representation**

Place/transition net



Arc-constant CPN



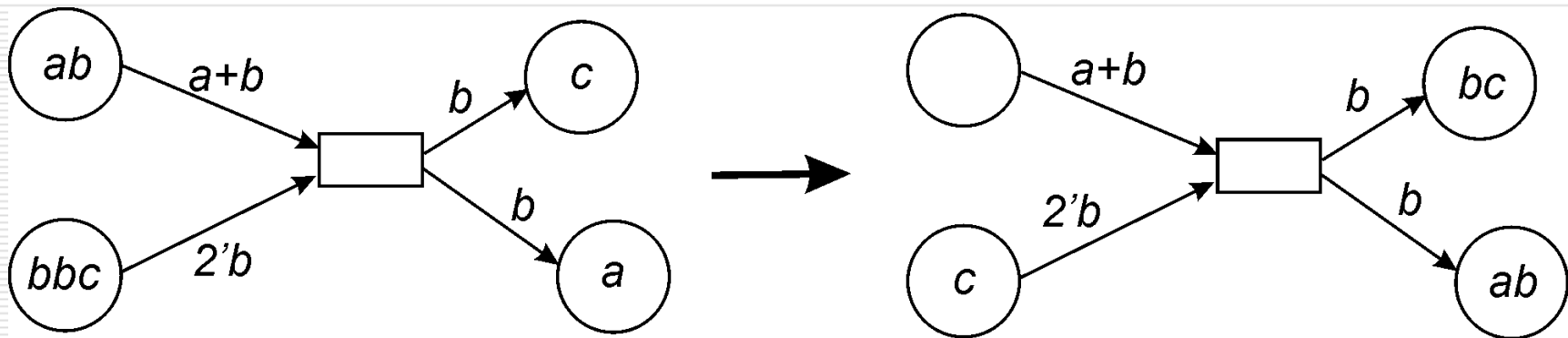
colour sets:

control = {*s*}

printers = {*A*, *B*}

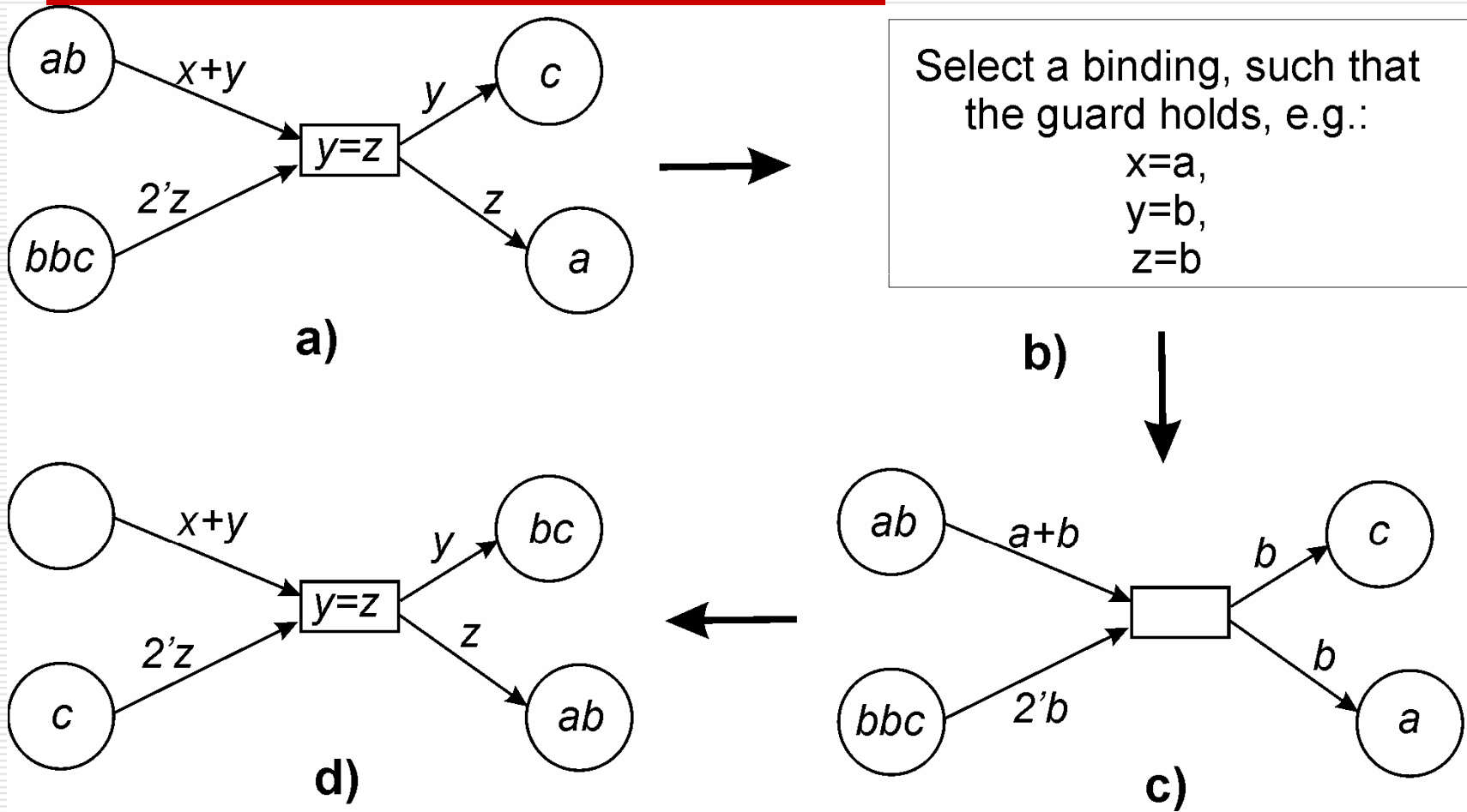
constants: *s*, *A*,

Firing rule for ac-CPN

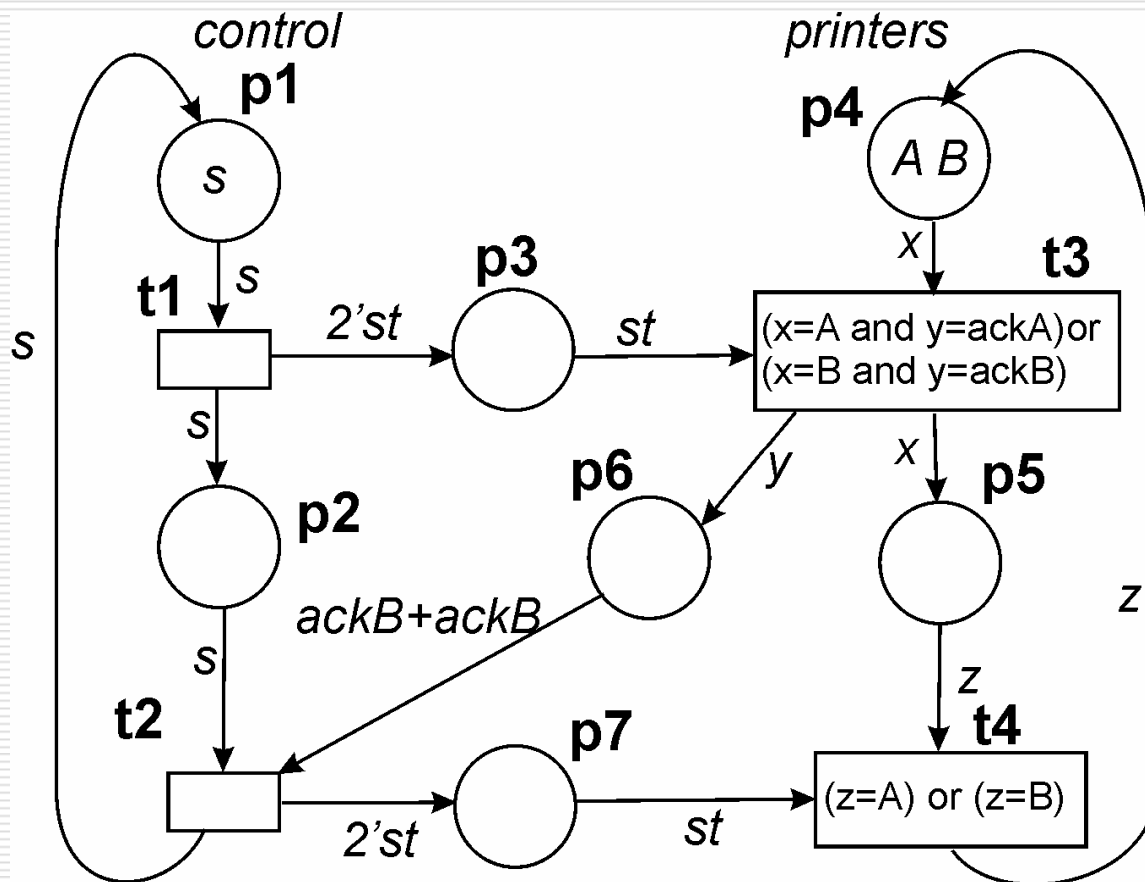


from ac-CPN to CPN ...

Firing rule for CPN



CPN model of printers



colour sets:

control = {*s*}

printers = {*A*, *B*}

ack = {*ackA*, *ackB*}

strobe = {*st*}

variables: *x*, *y*, *z*, *st*

constants: *ackA*, *ackB*,
stA, *stB*

Content of a course:

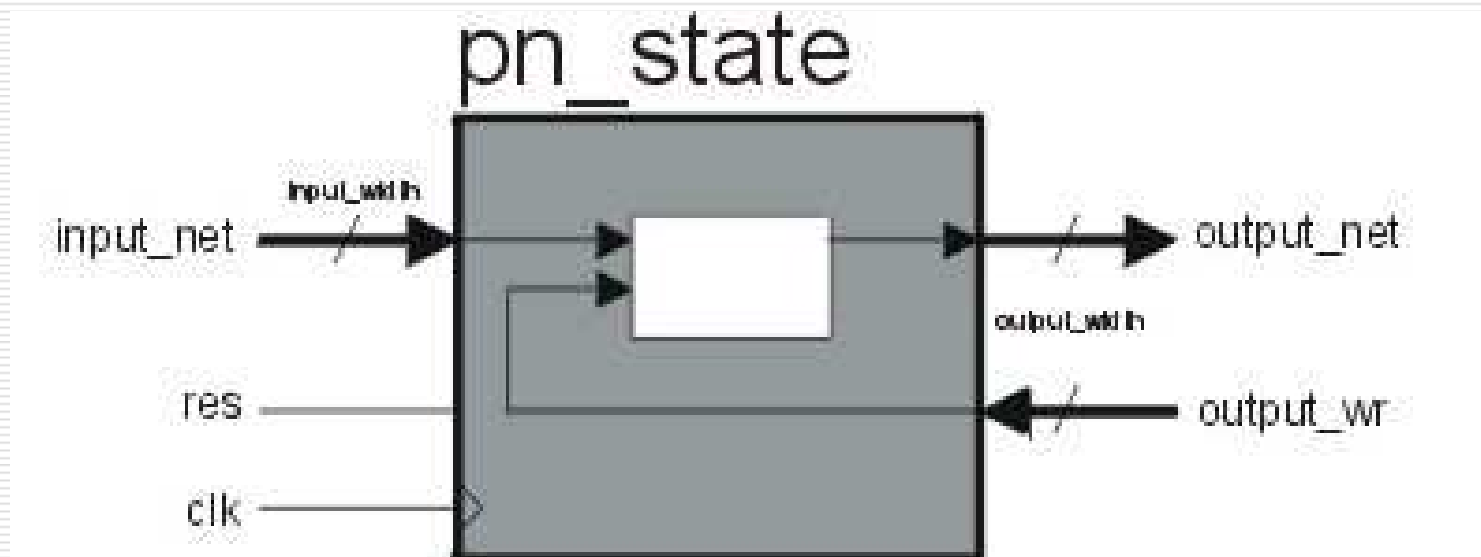
- seminars (6)
- labs (1)
- group projects (from 3 to 5 students):
 - case study (web browsing)
 - software understanding and using
 - web pages construction
 - final presentation

all during 3 month

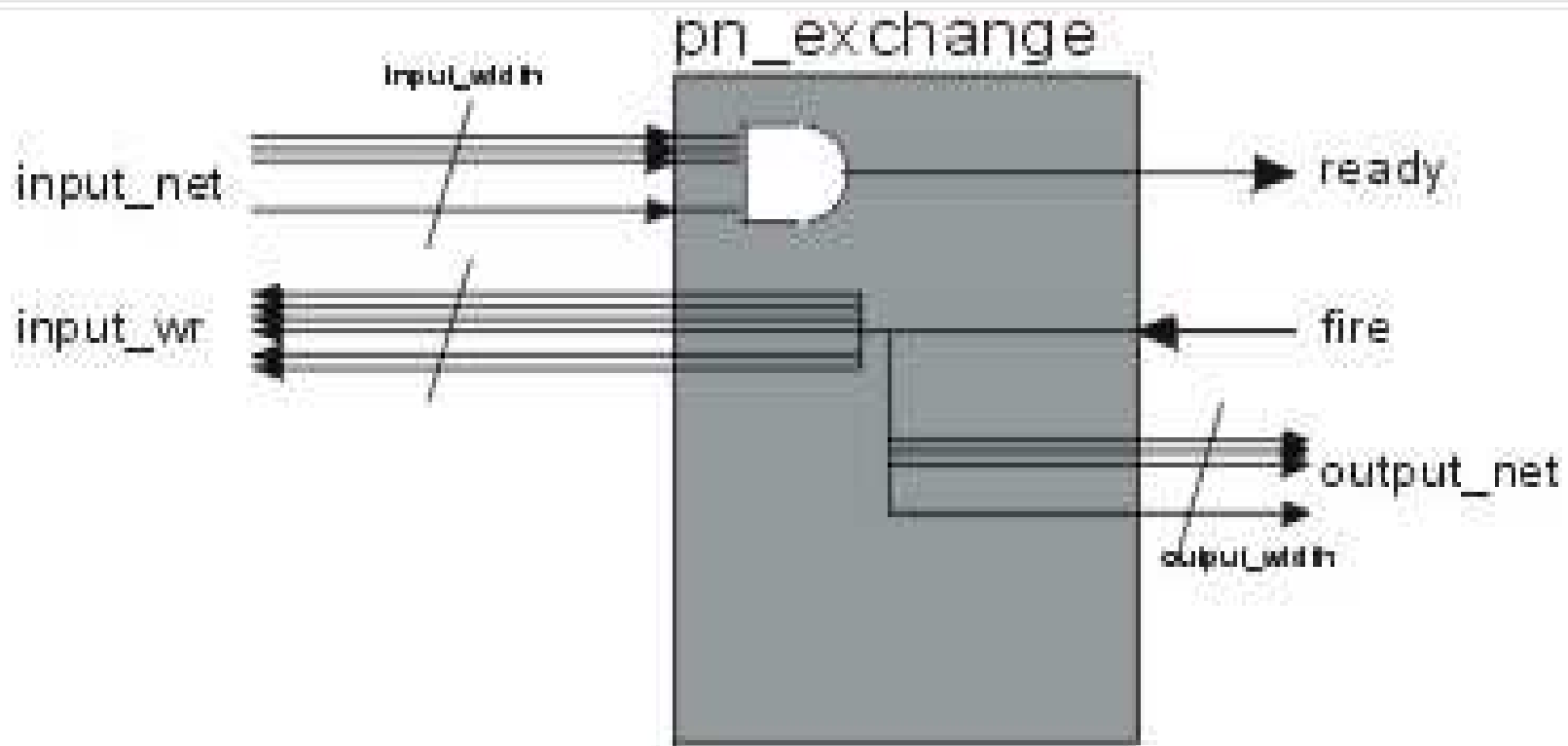
Petri Net Applications in Students Projects - limits

- ❑ direct implementation of the Petri net model in hardware (FPGA)
- ❑ using software tools (Design/CPN or JARP editor, CPN Tools
<http://wiki.daimi.au.dk/cpntools/cpntools.wiki>)
- ❑ unified description in PNML language
- transformed by our *pnml2vhdl* compiler to synthesized VHDL and then into the FPGA bitstream

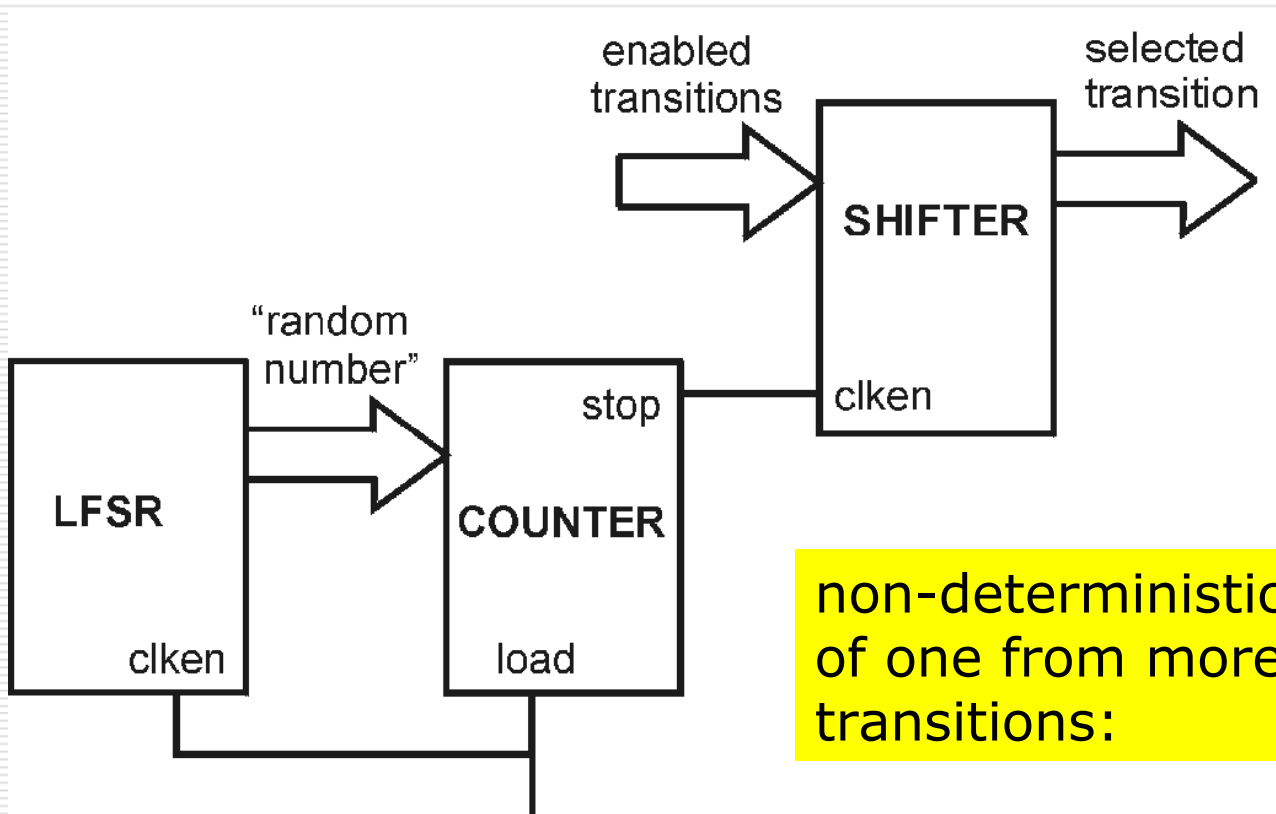
Place block



Transition block

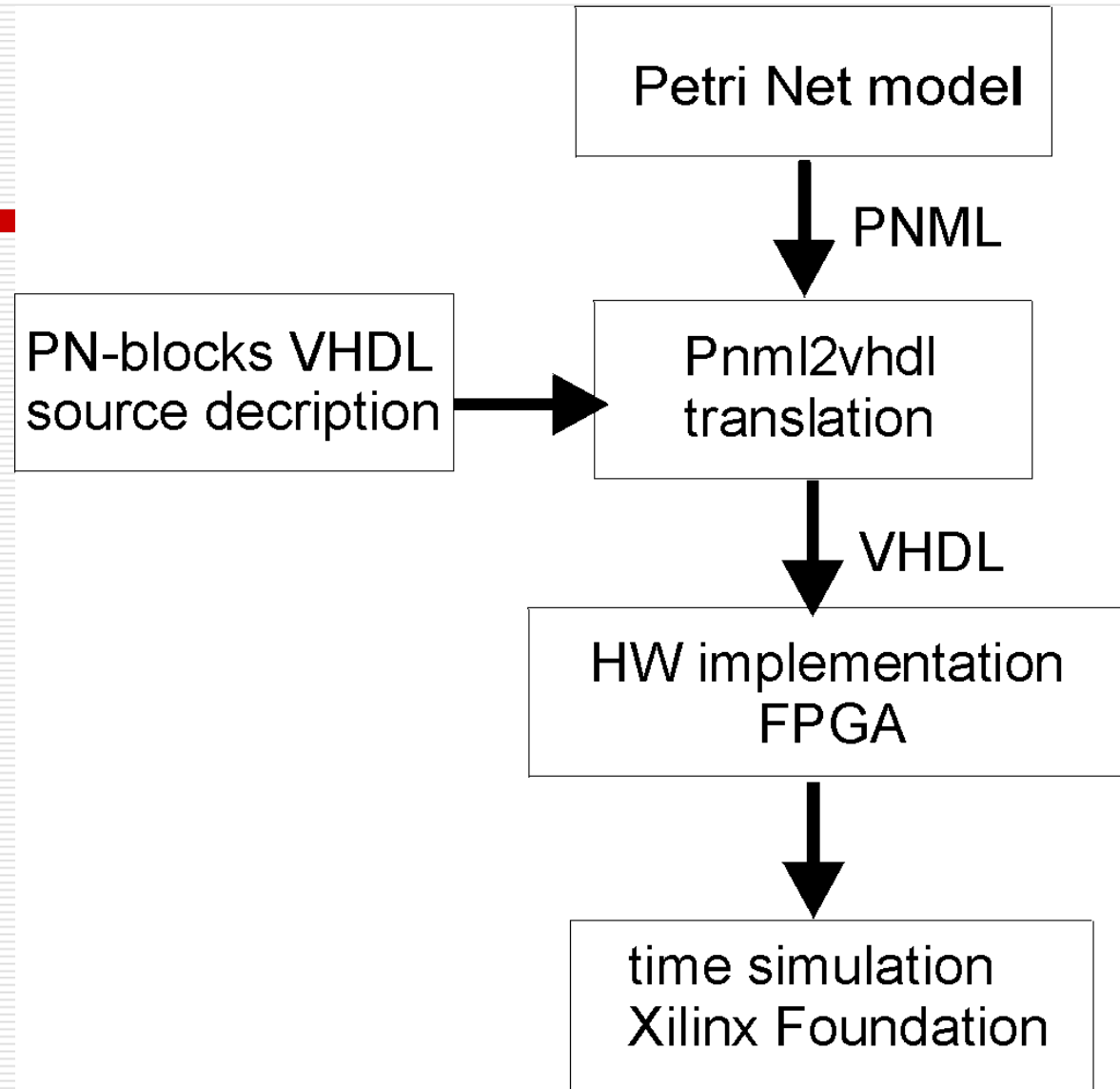


Random selection block

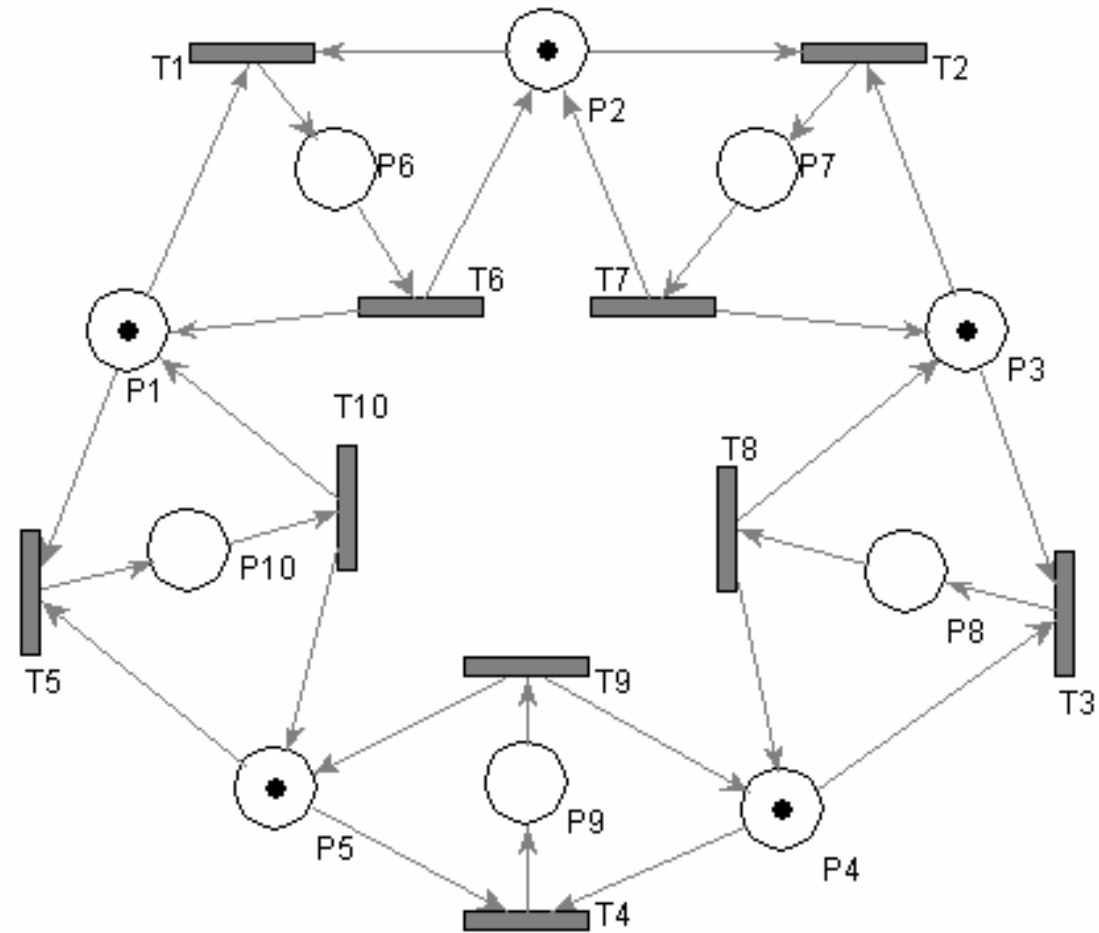


non-deterministic selection of one from more enabled transitions:

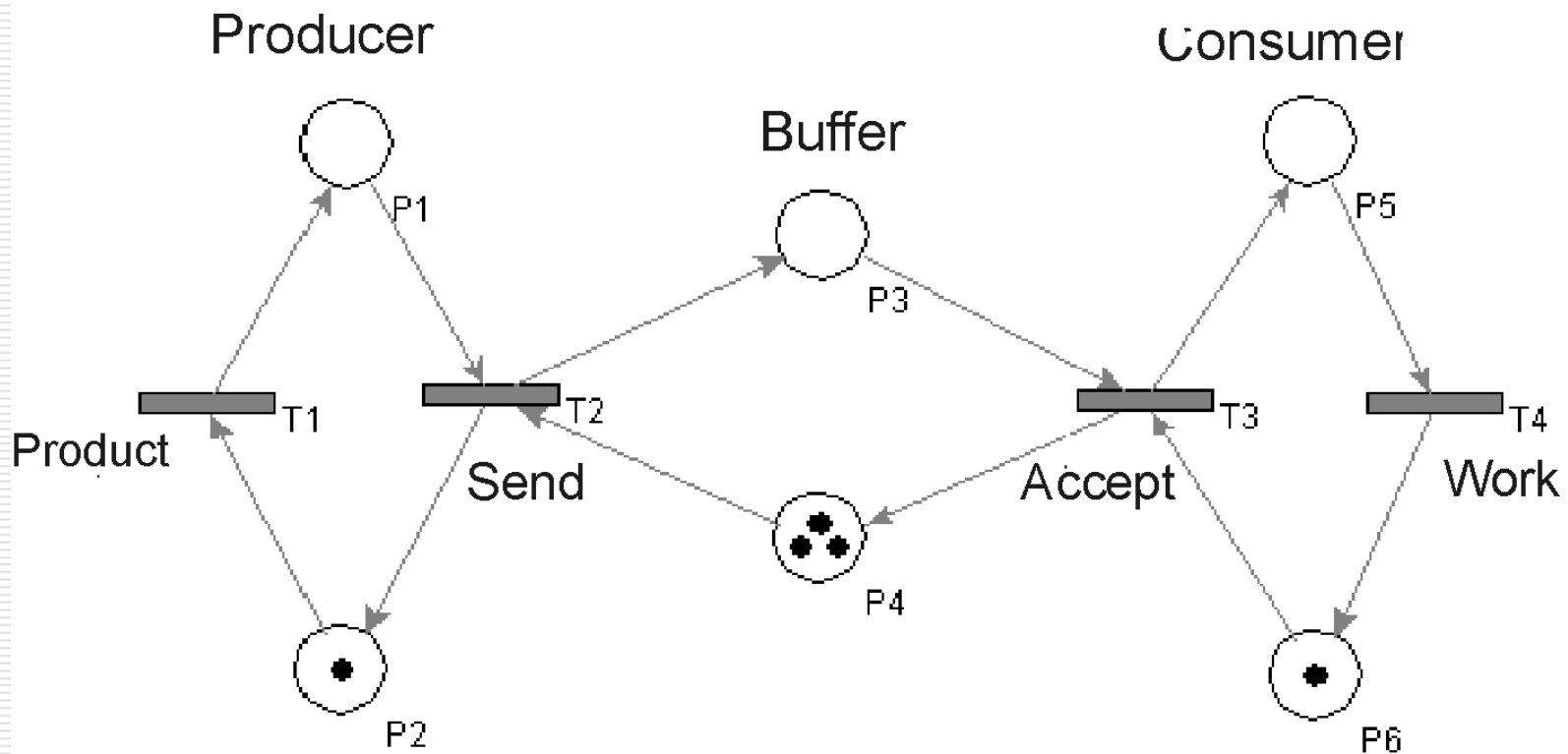
Algorithm description



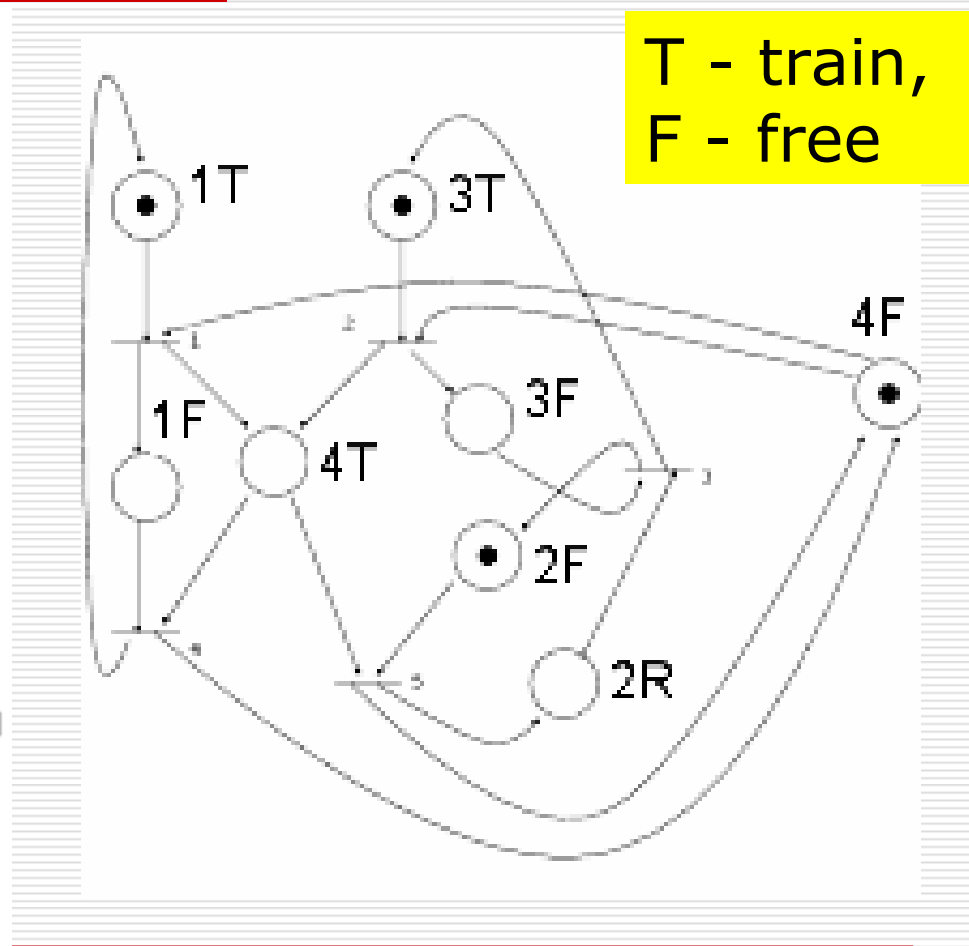
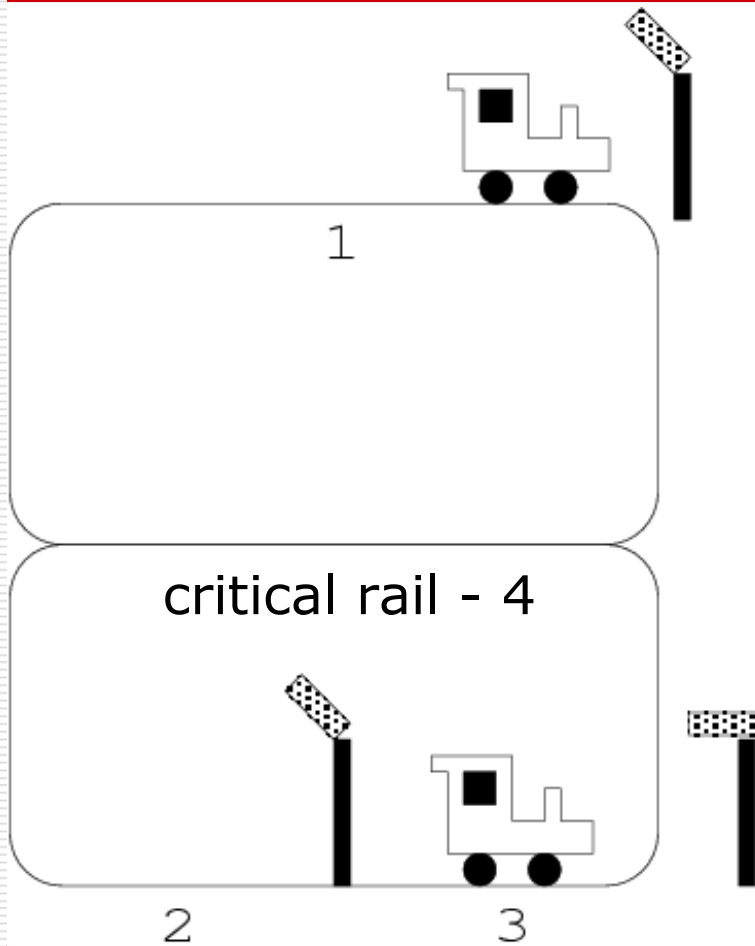
Real models - dinning philosophers



Producer – consumer model



Railway semaphore model



Problems

- ❑ real models are complicated
- ❑ new student's group has to understand to new SW tools, to older project, ...
- ❑ the greatest one - what to model?
- ❑ many projects are Petri net-models with "something more", with special interpretation, with new elements or properties, ...

Conclusions and future

- new models and their direct implementation with respect to some optimization (space, working frequency, equivalency with FSM)
- using such models with PNML description that can be translated into hardware (directly)
- hardware/software codesign
- fault-tolerant design

.....

questions, experiences, ...

