Hardware Systems Representations

Paolo PRINETTO

Director
CINI Cybersecurity
National Laboratory

Paolo.Prinetto@polito.it

Mob. +39 335 227529

CYBER CHALLENGE
CyberChallenge.IT

cini
Cybersecurity
National Lab

https://cybersecnatlab.it

License & Disclaimer

License Information

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Goal

- The lecture aims at presenting the *Representation domains* and the *Abstraction levels* most widely used to represent the design steps of a digital system.

Prerequisites

- Lecture:
  - HW_S_1 – Hardware Systems Definitions & Taxonomies Part I & II

Further readings

- Students interested in making a reference to a textbook on the arguments covered in this lecture can refer, for instance, to:
  - G. Conte, A. Mazzeo, N. Mazzocca, P. Prinetto: “Architettura dei calcolatori”, Città Studi, 2015 (Chapter 1: Classificazioni e Concetti base) (In Italian)

Outline

- Representation matrix
- Behavioral domain
- Structural domain
- Physical domain
- Design evolution

Outline

- Representation matrix
- Behavioral domain
- Structural domain
- Physical domain
- Design evolution

Managing complexity

- To manage and to tackle its complexity, a system must be described resorting to several *abstraction layers*

Abstraction

- An abstraction is a simplified model of the system, showing only the selected features and ignoring the associated details

Managing complexity

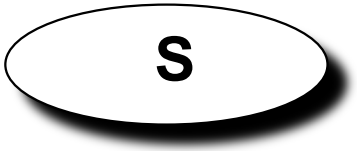
- To manage and to tackle its complexity, a system must be described resorting to several *abstraction layers*
- In addition, at each layer, specific subsets of aspects of interest must be considered

The sub-space abstraction ↔ domain

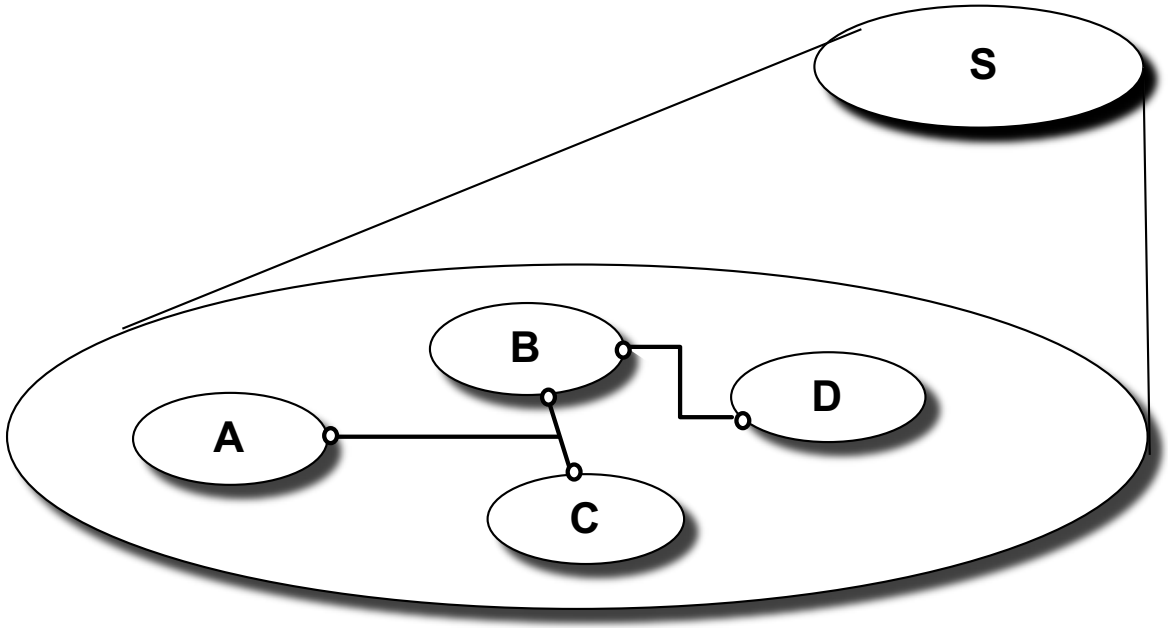
From High to Low

- A high-level abstraction contains only the most vital data
- A low-level abstraction is more detailed and takes account of previously ignored information
- Although it is more complex, a low-level abstraction model is more accurate and is closer to the real circuit.

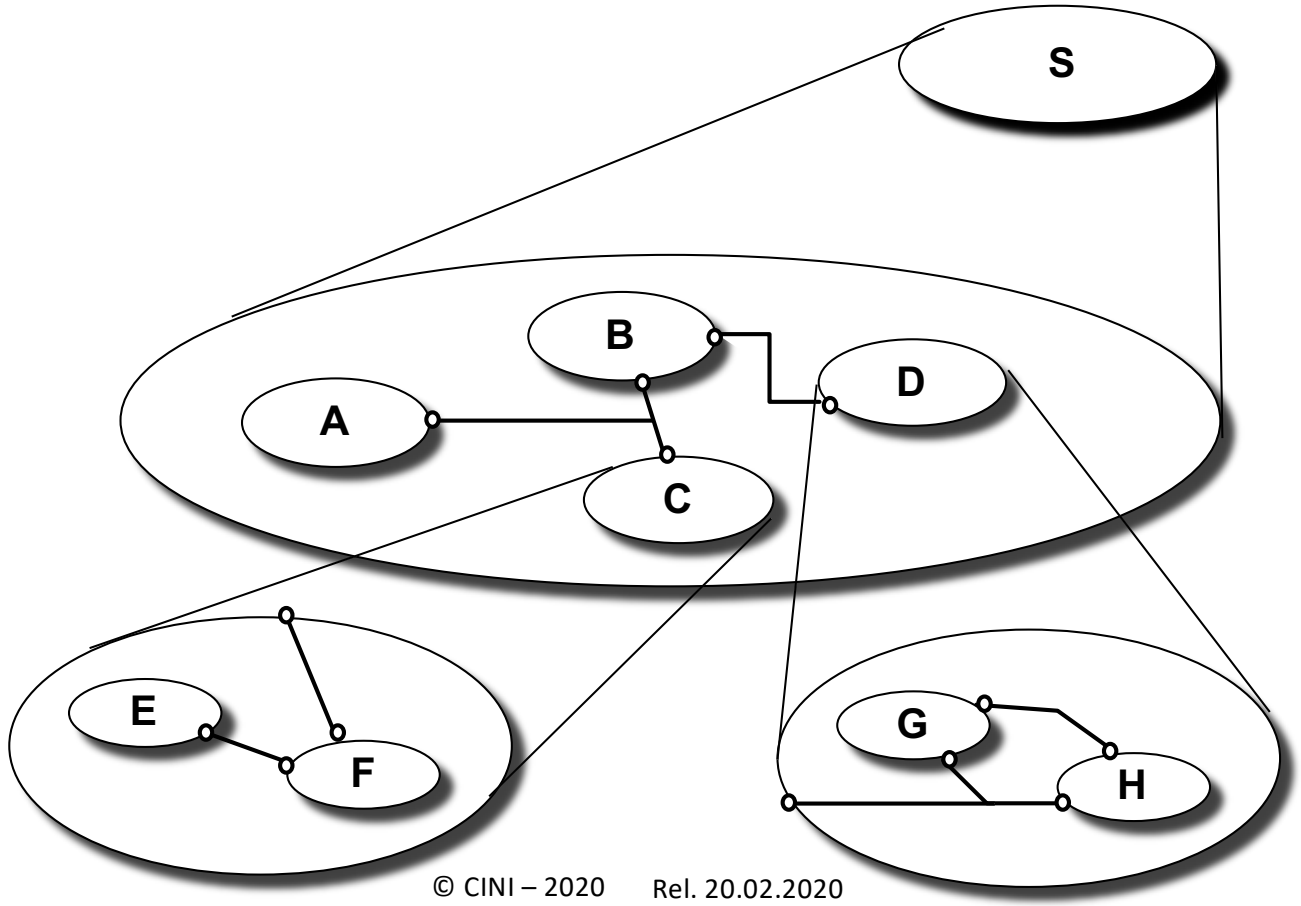
Example of Hierarchical decomposition



Example of Hierarchical decomposition



Example of Hierarchical decomposition

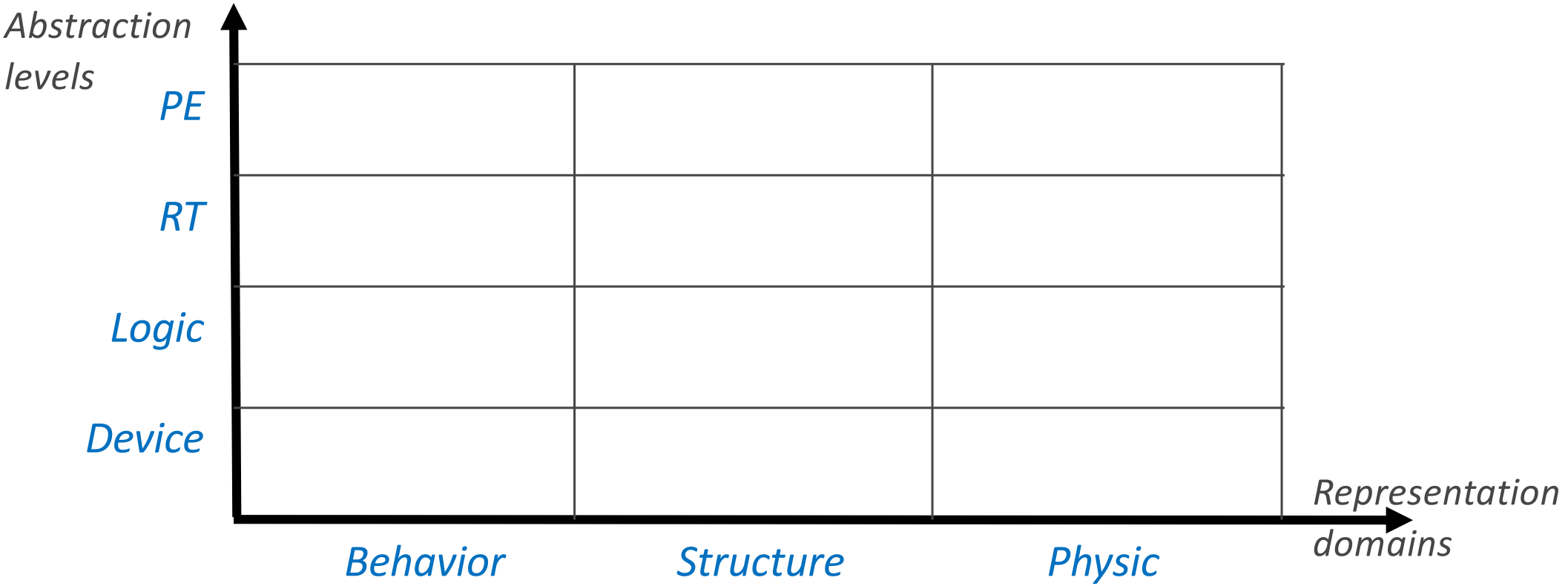


 Rel. 20.02.2020

Design representations

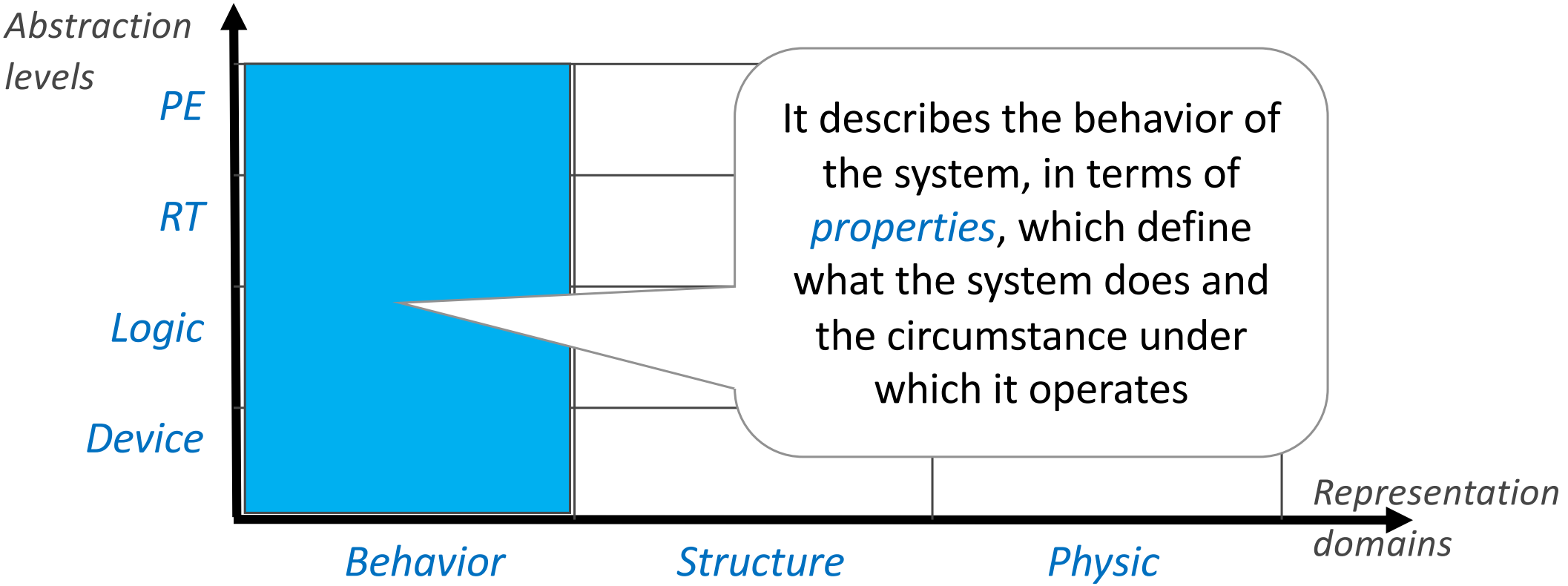
- The output of each design step must be “saved” or “represented” properly.
- Due to the system’s complexity, there in no “one-size-fits-all” efficient solutions

Representation Matrix



 Rel. 20.02.2020

Representation Matrix



 Rel. 20.02.2020

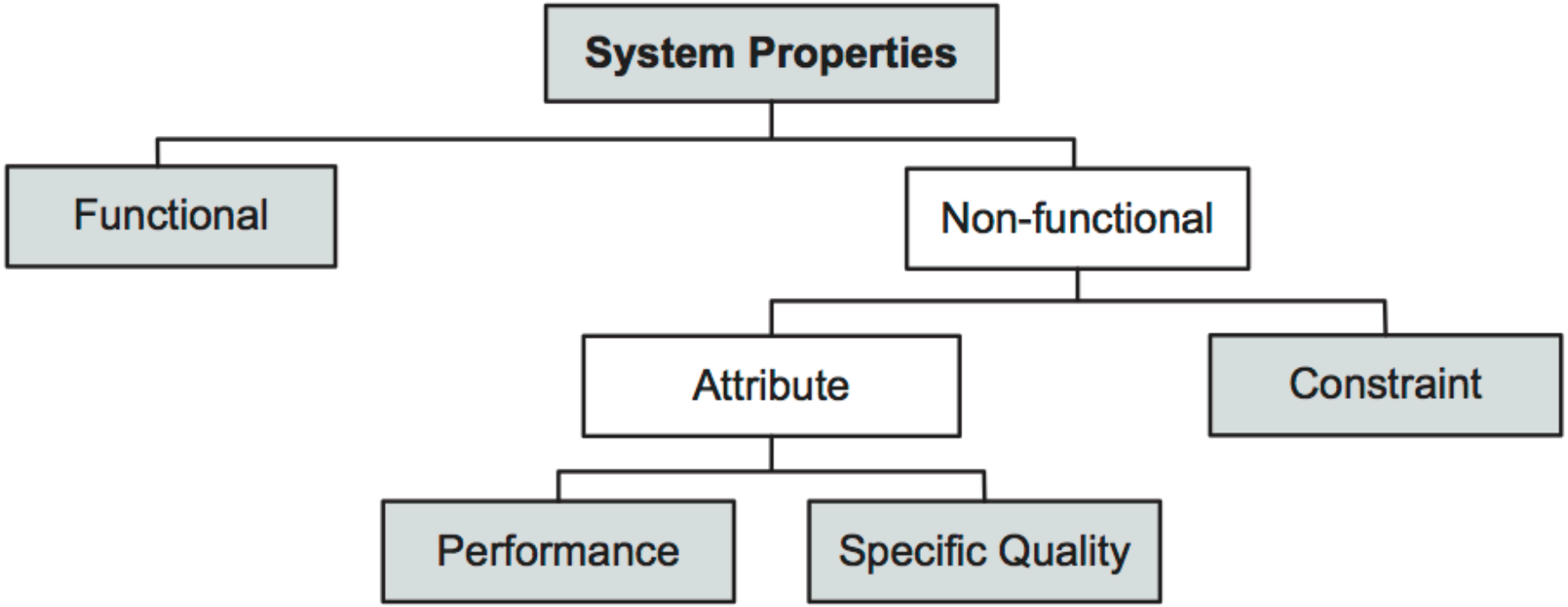
System Properties

Functional

- A property that specifies:
  - the inputs (stimuli) to the system
  - the outputs (responses) from the system
  - the behavioral relationships between them

Non-Functional

- A property that specifies:
  - an attribute, i.e., a performance specification or a specific quality of the system, or
  - a constraint on the system



Functionality and behavior:

- Functions
- Data
- Stimuli
- Behavior
- ...

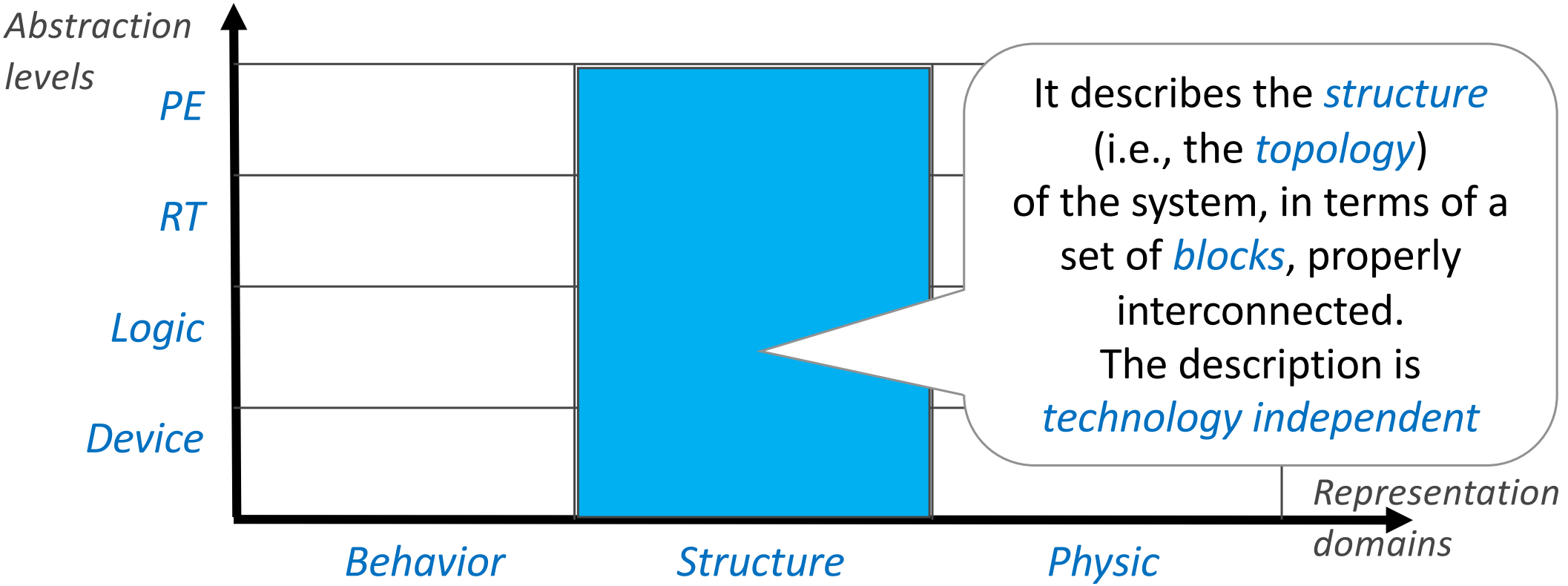
- Timing
- Delay
- Area
- Power
- Temperature
- ...

"-ilities":

- Reliability
- Vulnerability
- Usability
- Security
- Availability
- ...

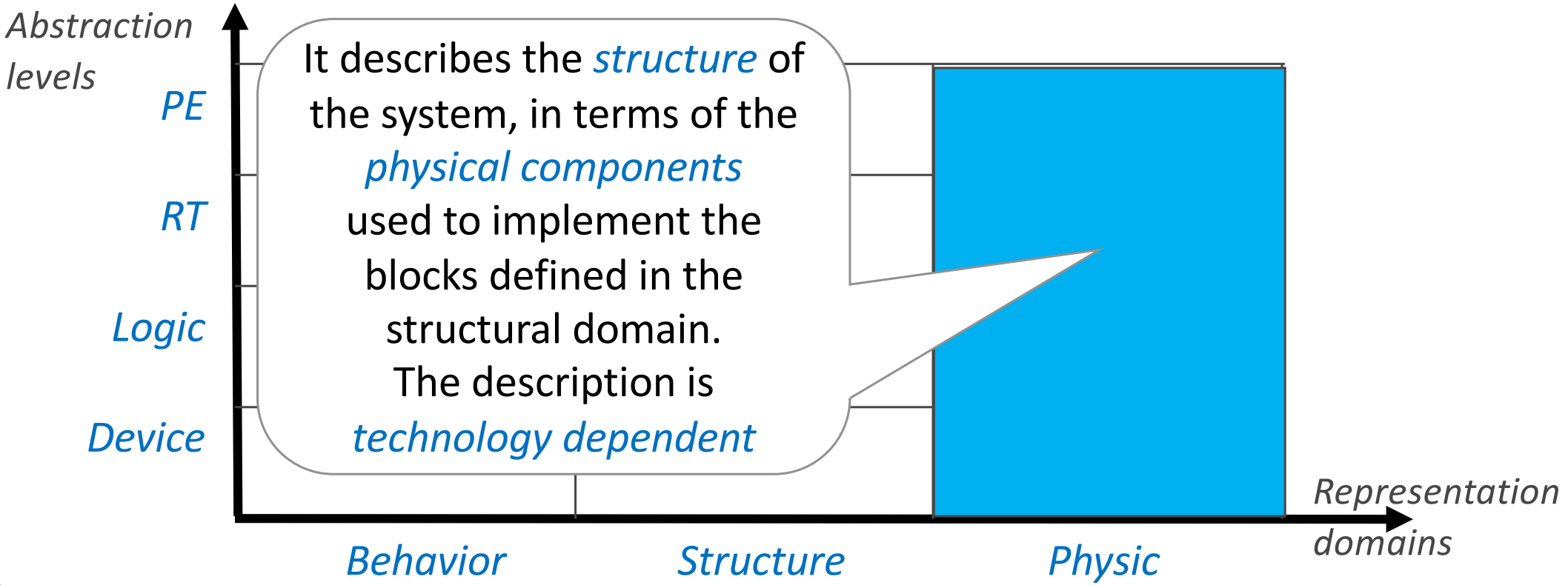
- Physical
- Legal
- Cultural
- Environmental
- Design and Implementation
- ...

Representation Matrix



 Rel. 20.02.2020

Representation Matrix

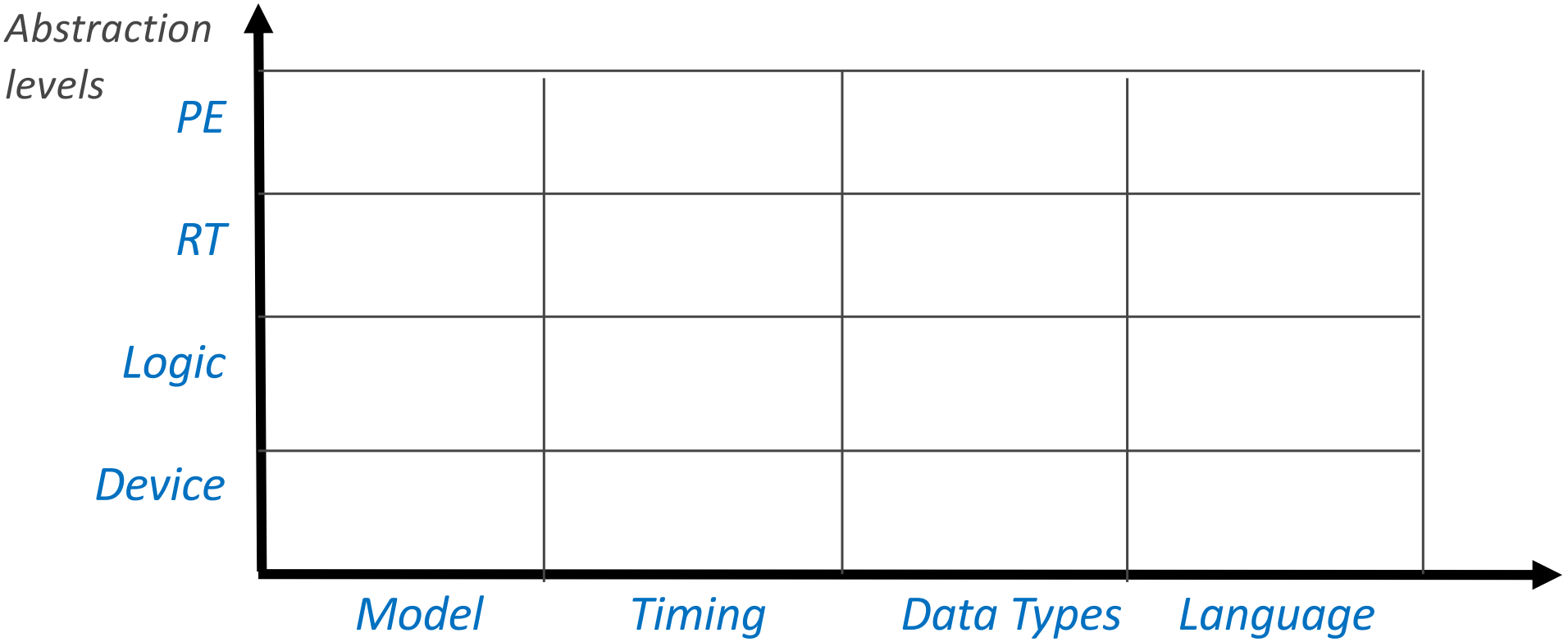


 Rel. 20.02.2020

Outline

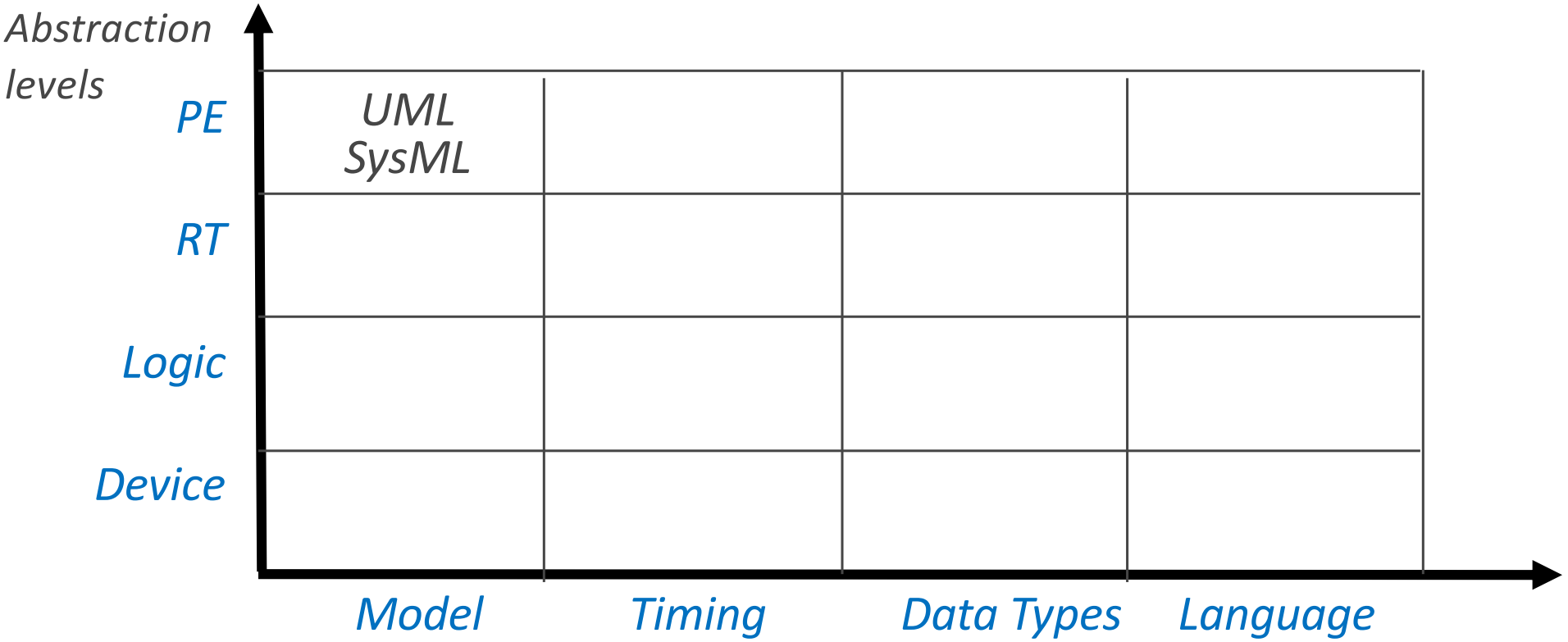
- Representation matrix
- Behavioral domain
- Structural domain
- Physical domain
- Design evolution

Behavioral domain



 Rel. 20.02.2020

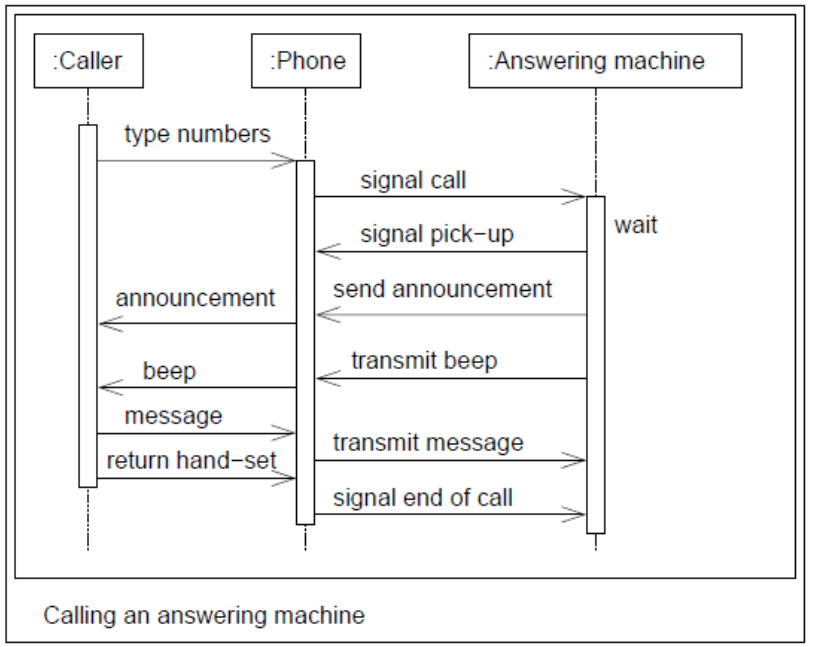
Behavioral domain – PE level



Examples: UML (Unified Modeling Language)

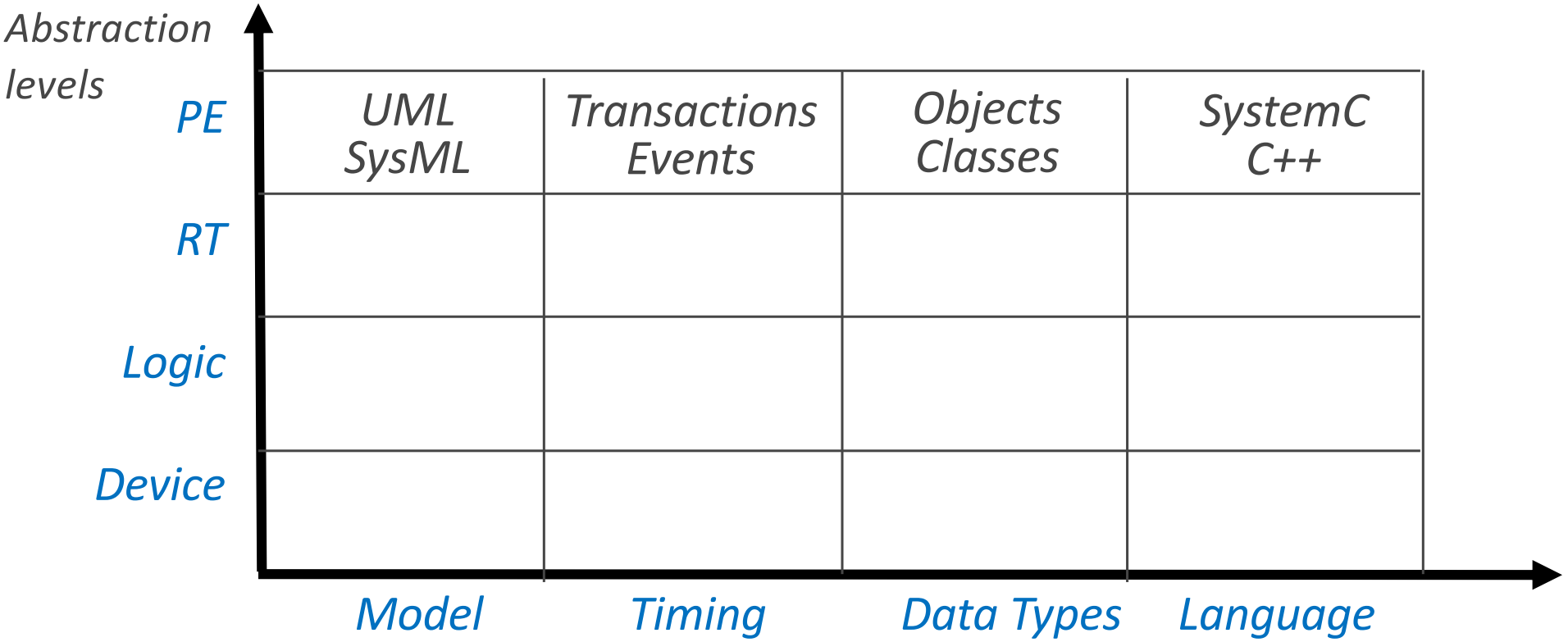
Use cases

Sequence charts



 Rel. 20.02.2020

Behavioral domain – PE level

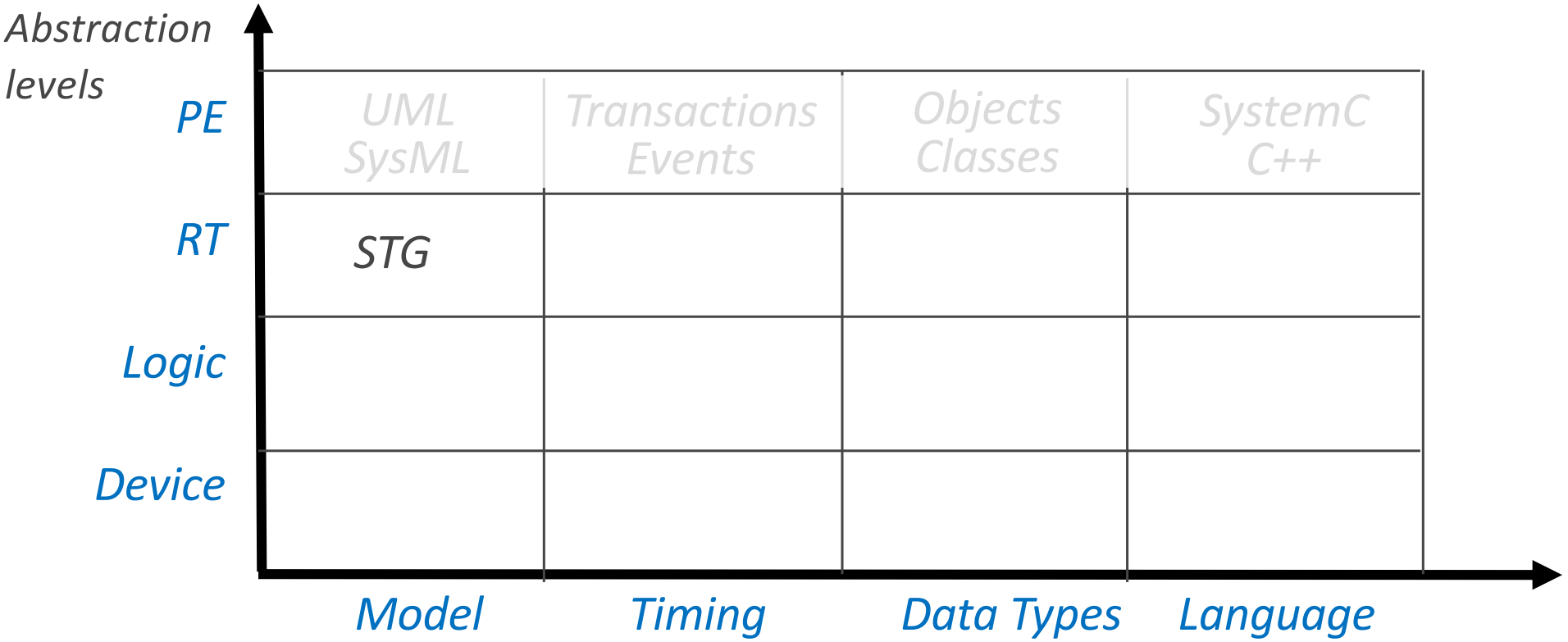


Examples: C++ description

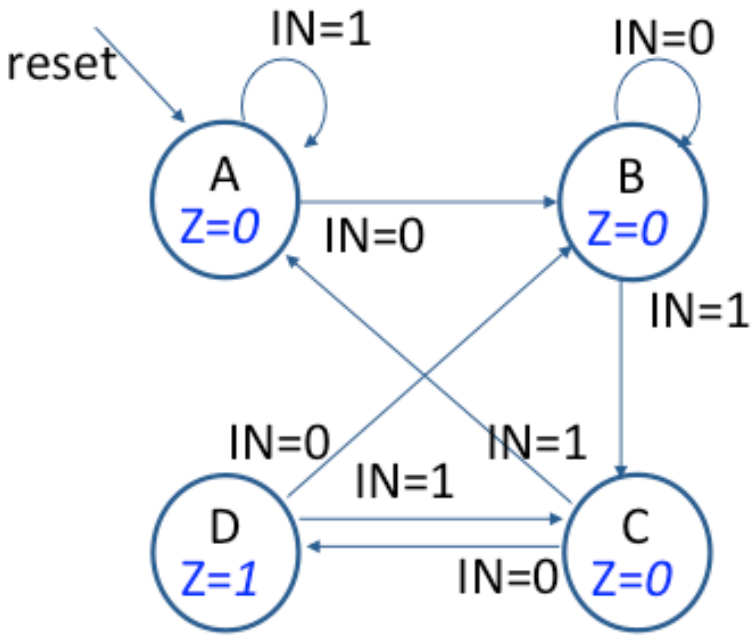

```
#include<limits.h>
unsigned int ex (unsigned int a,
                 unsigned int b,
                 bool *ovrfl,
                 bool max_add_n) {
  unsigned int q;
  if ( max_add_n ) { //Max
    *ovrfl = false;
    c = max(a, b);
  }
  else { //Add
    c = add(a, b, ovrfl);
  }
  return(c);
}
```

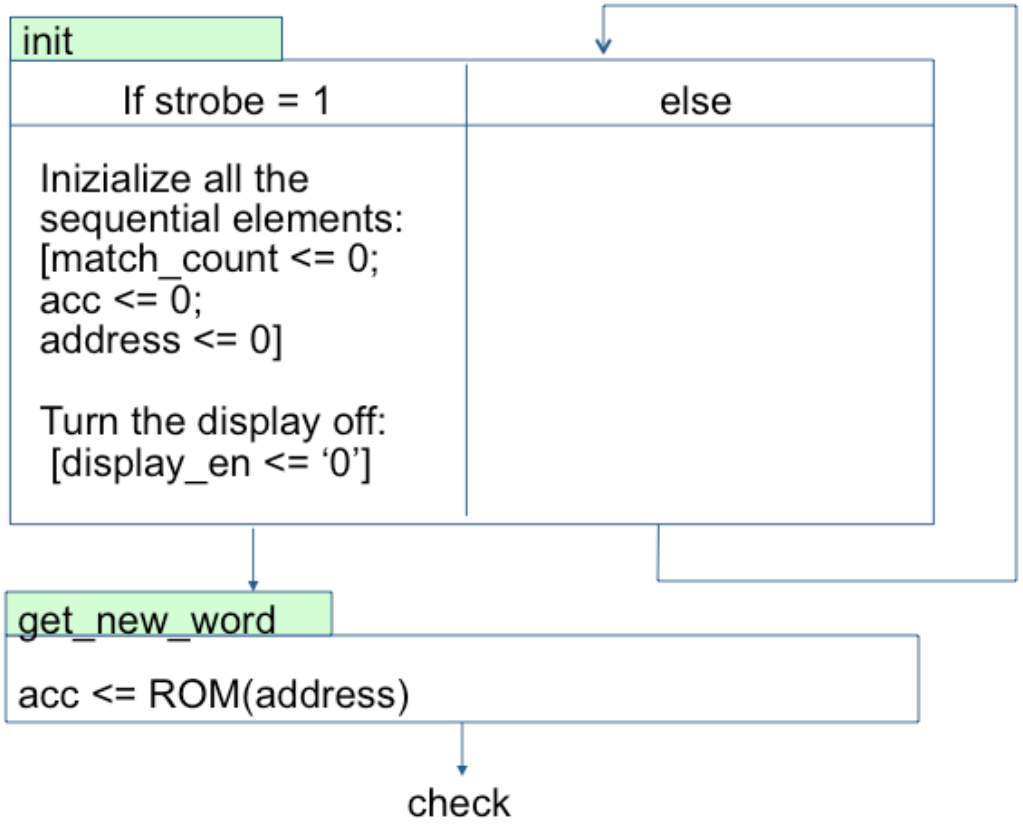
```
unsigned int add (unsigned int a_var,
                  unsigned int b_var,
                  bool *ovrfl) {
  unsigned int sum;

  if (a_var > UINT_MAX - b_var) {
    *ovrfl = true;
  }
  else {
    *ovrfl = false;
    sum = a_var + b_var;
  }
  return(sum);
}
```

Behavioral domain – RT level

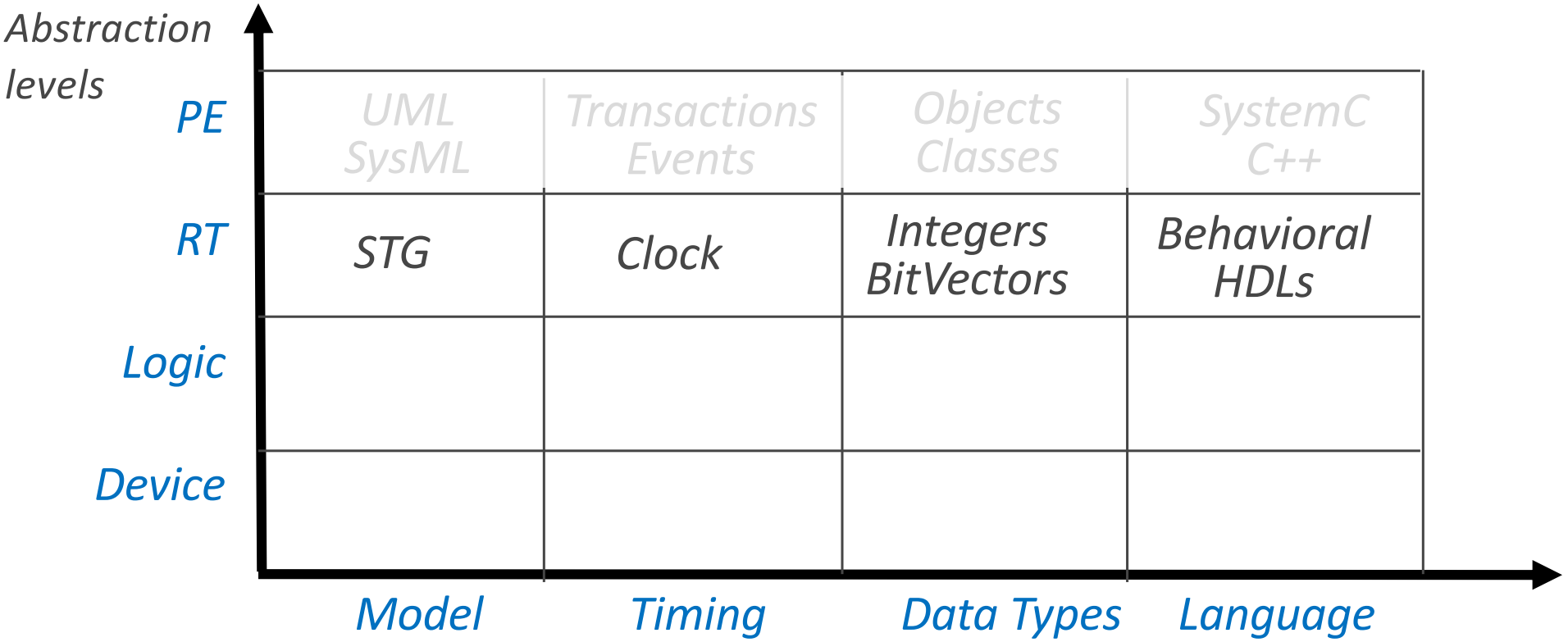


Examples: State Transition Graph (STG)





Behavioral domain – RT level

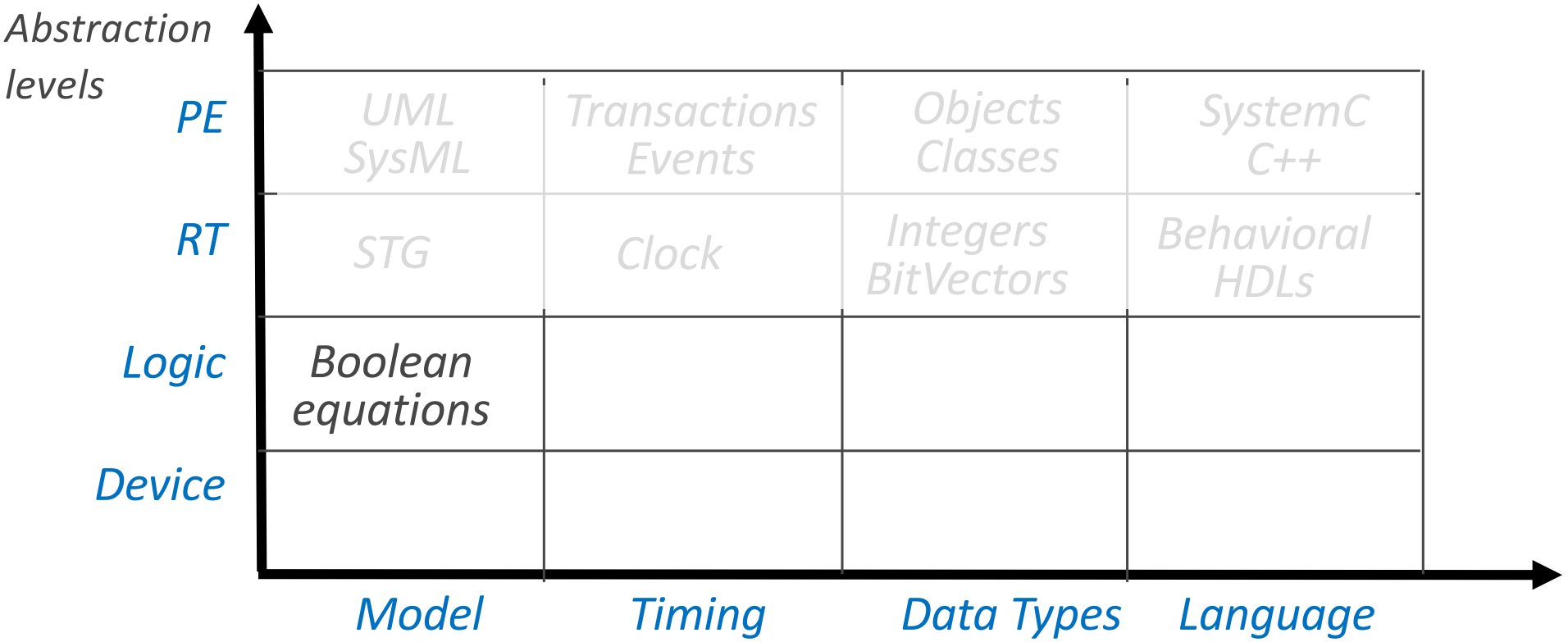


Examples: Behavioral VHDL


```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity ex is
generic (N: integer:=32);
port(
   clk            : in std_logic;
   rst            : in std_logic;
   a              : in std_logic_vector(N-1 downto 0);
   b              : in std_logic_vector(N-1 downto 0);
   max_add_n      : in std_logic;
   c              : out std_logic_vector(N-1 downto 0);
   ovrfl          : out std_logic_vector(0 downto 0));
end entity ex;
```

```
architecture behavioural of ex is
signal a_reg : std_logic_vector(N downto 0);
signal b_reg : std_logic_vector(N downto 0);
begin
sync_proc: process (clk, rst)
 begin
  if rst = '1' then
    c    <= (others => '0');
    a_reg <= (others => '0');
    b_reg <= (others => '0');
    ovrfl <= (others => '0');
  elsif (clk'event and clk = '1') then
    a_reg(N-1 downto 0) <= a;
    b_reg(N-1 downto 0) <= b;
    if ( max_add_n = '1' ) then -- MAX
      ovrfl <= (others => '0');
      if ( (unsigned(a_reg)) > (unsigned(b_reg)) ) then
        c(N-1 downto 0) <= a_reg(N-1 downto 0);
      else
        c(N-1 downto 0) <= b_reg(N-1 downto 0);
      end if;
    else -- ADD
      c    <= conv_std_logic_vector(unsigned(a_reg) +
              unsigned(b_reg),N+1)(N-1 downto 0);
      ovrfl <= conv_std_logic_vector(unsigned(a_reg) +
               unsigned(b_reg),N+1)(N downto N);
    end if;
  end if;
 end process sync_proc;
end behavioural;
```

Behavioral domain – Logic level



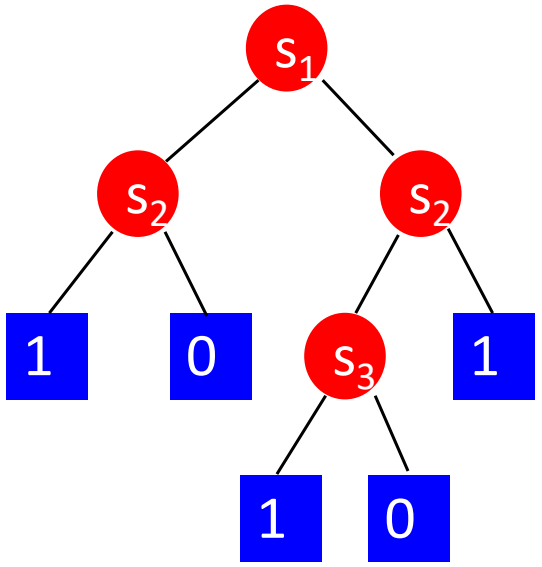
 Rel. 20.02.2020

Examples: Boolean equations

s_3 \ s_1 s_2	00	01	11	10
0	1	0	1	1
1	1	0	1	0

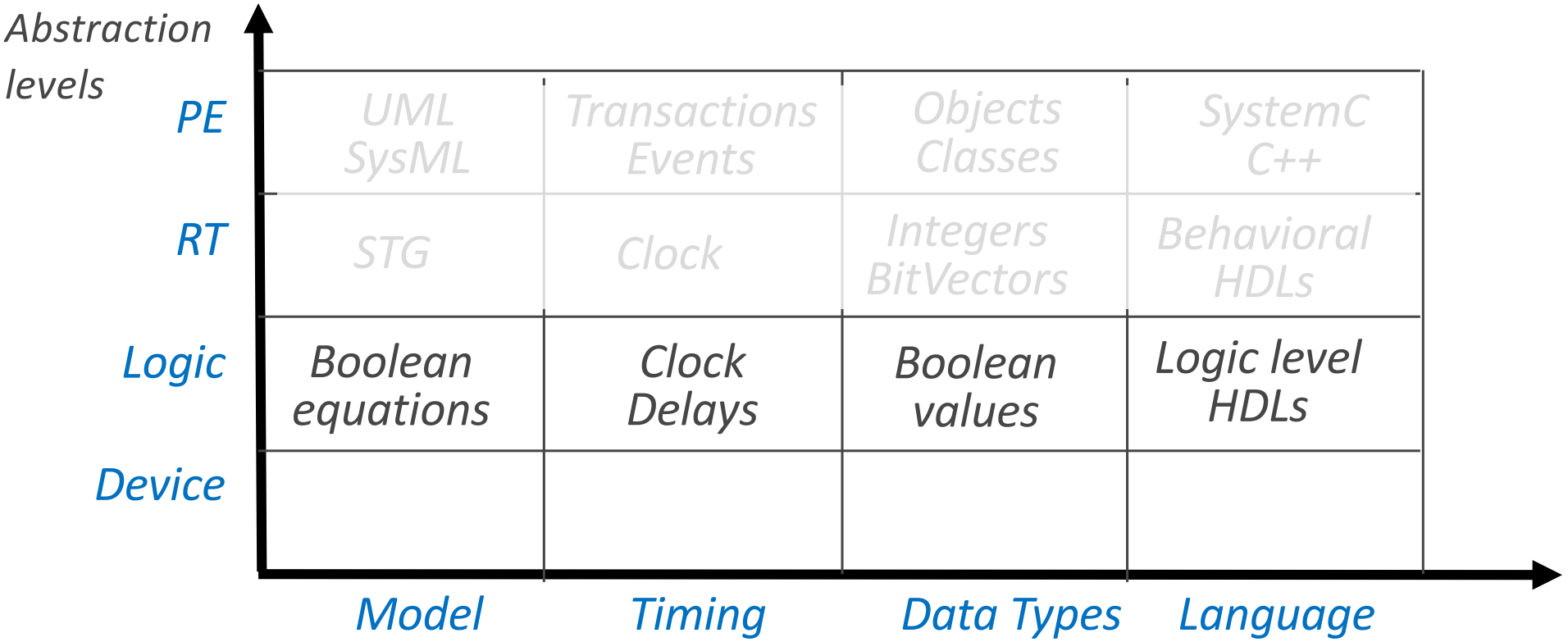
```
.i 3
.o 1
00- 1
1-0 1
11- 1
.e
```

$$s_1' s_2' + s_1 s_3' + s_1 s_2$$

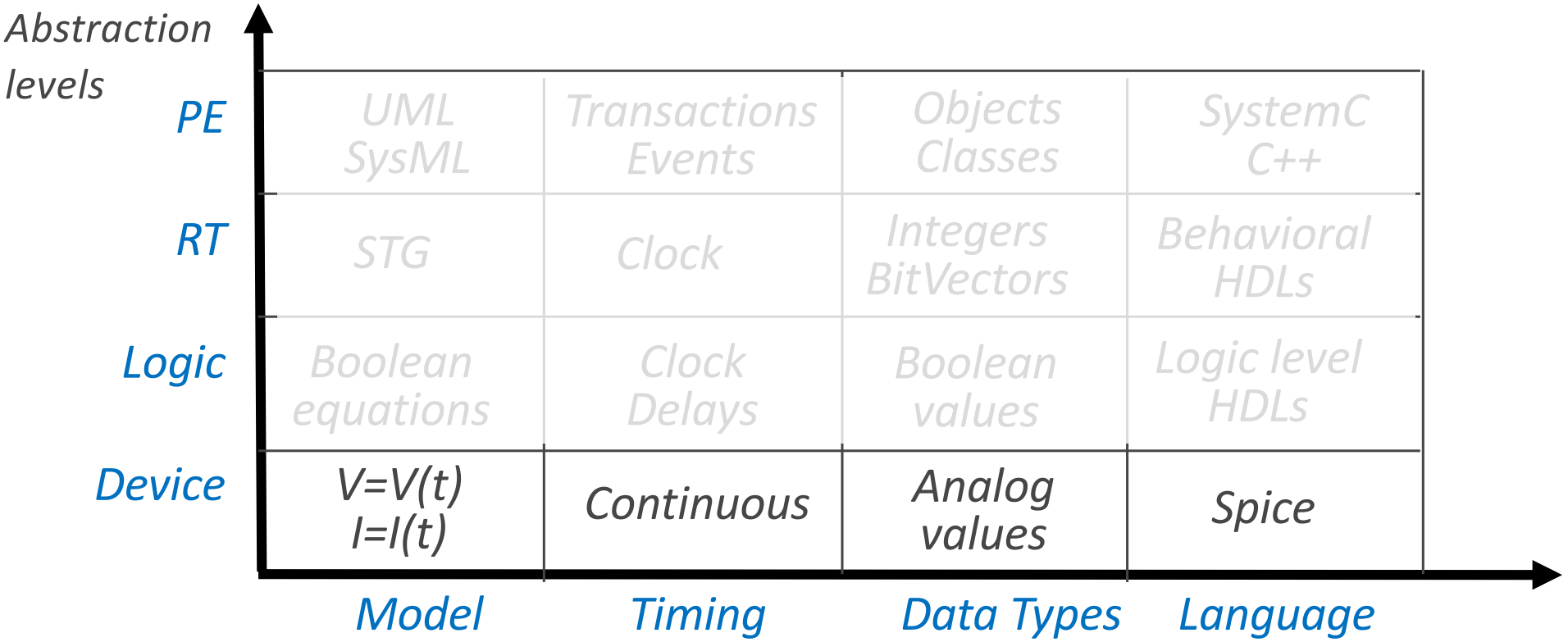


 Rel. 20.02.2020

Behavioral domain – Logic level



Behavioral domain – Device level

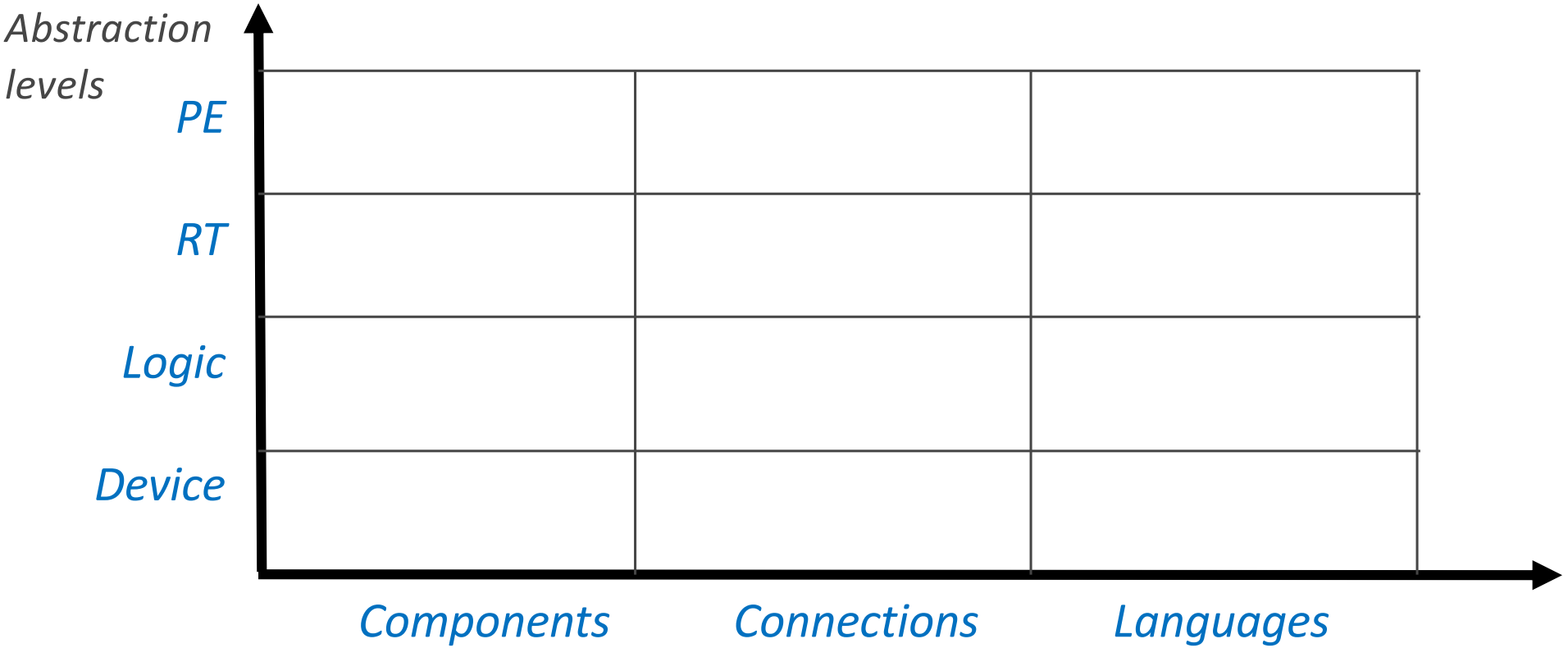


 Rel. 20.02.2020

Outline

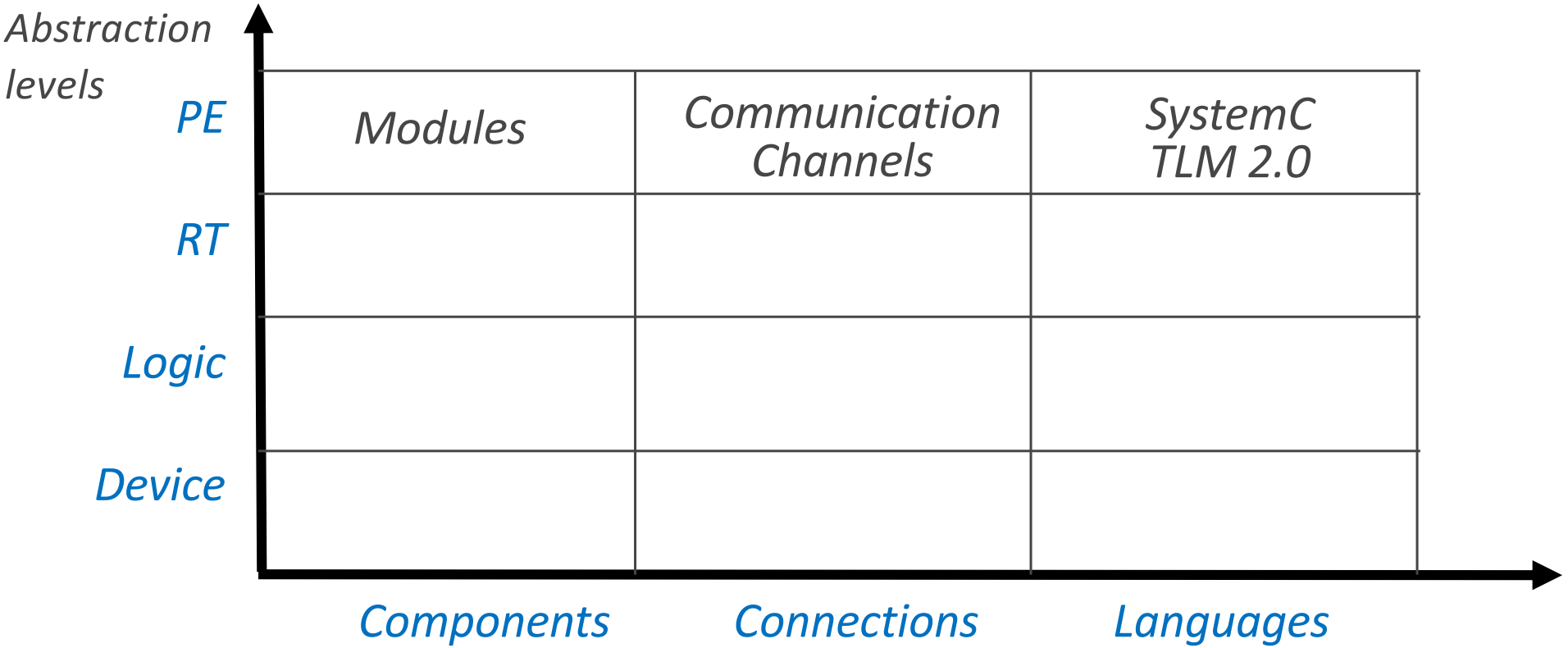
- Representation matrix
- Behavioral domain
- Structural domain
- Physical domain
- Design evolution

Structural Domain



 Rel. 20.02.2020

Structural domain – PE level

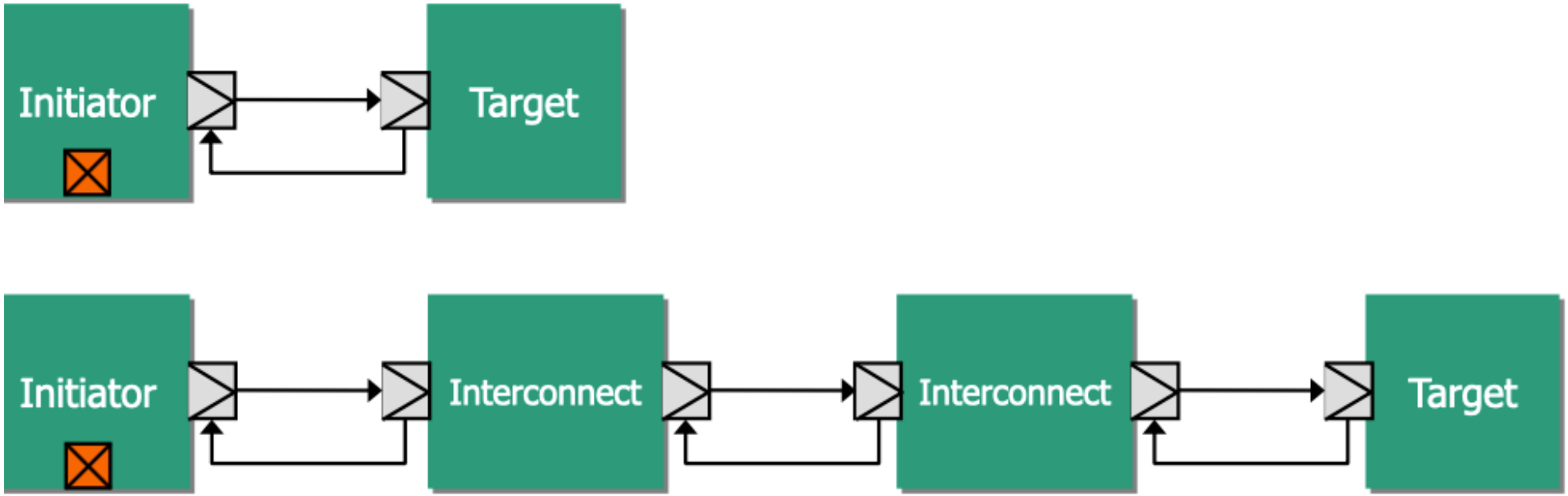


 Rel. 20.02.2020

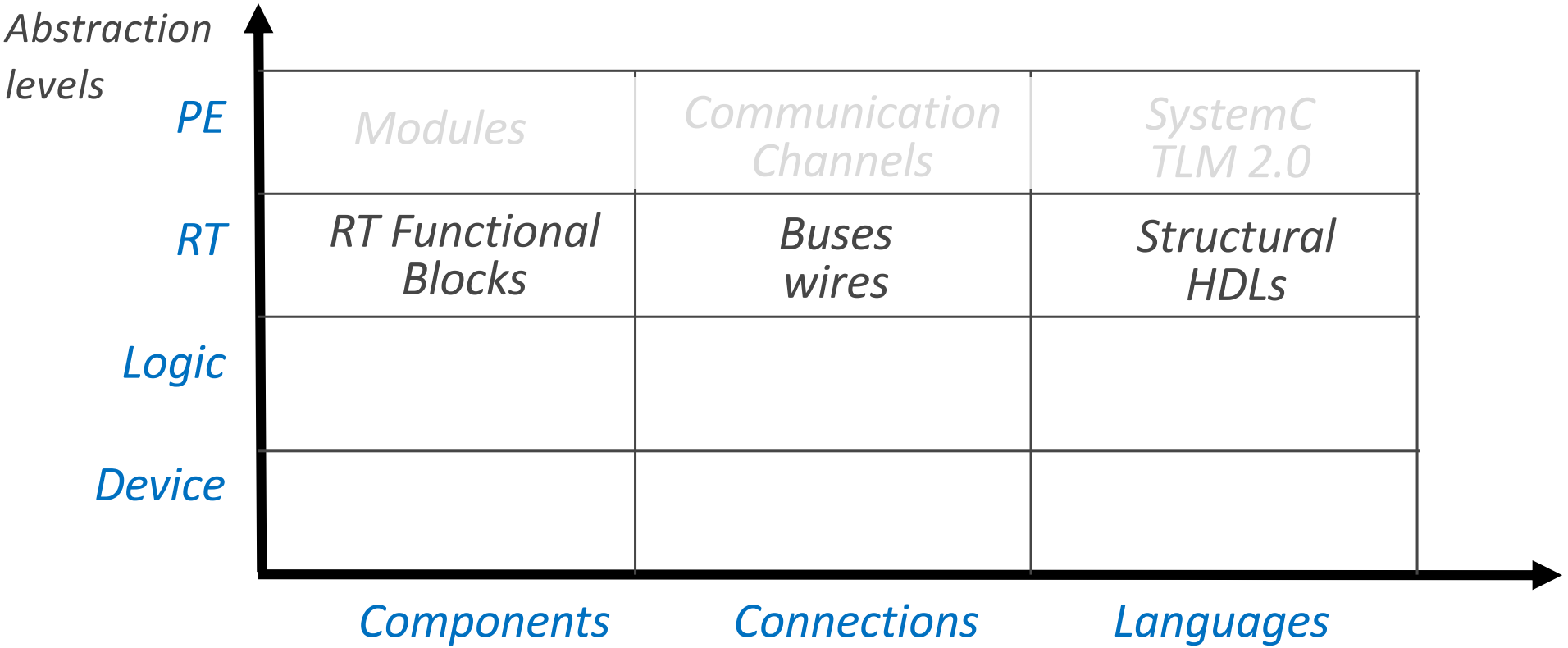
OCSI TLM 2.0 – Transaction-level modeling

- Transaction-level modeling is a high-level approach to modeling digital systems where details of communication among modules are separated from the details of the implementation of the functional units or of the communication architecture.
- It’s a standard of the Open SystemC Iniziative

Examples: TLM 2.0

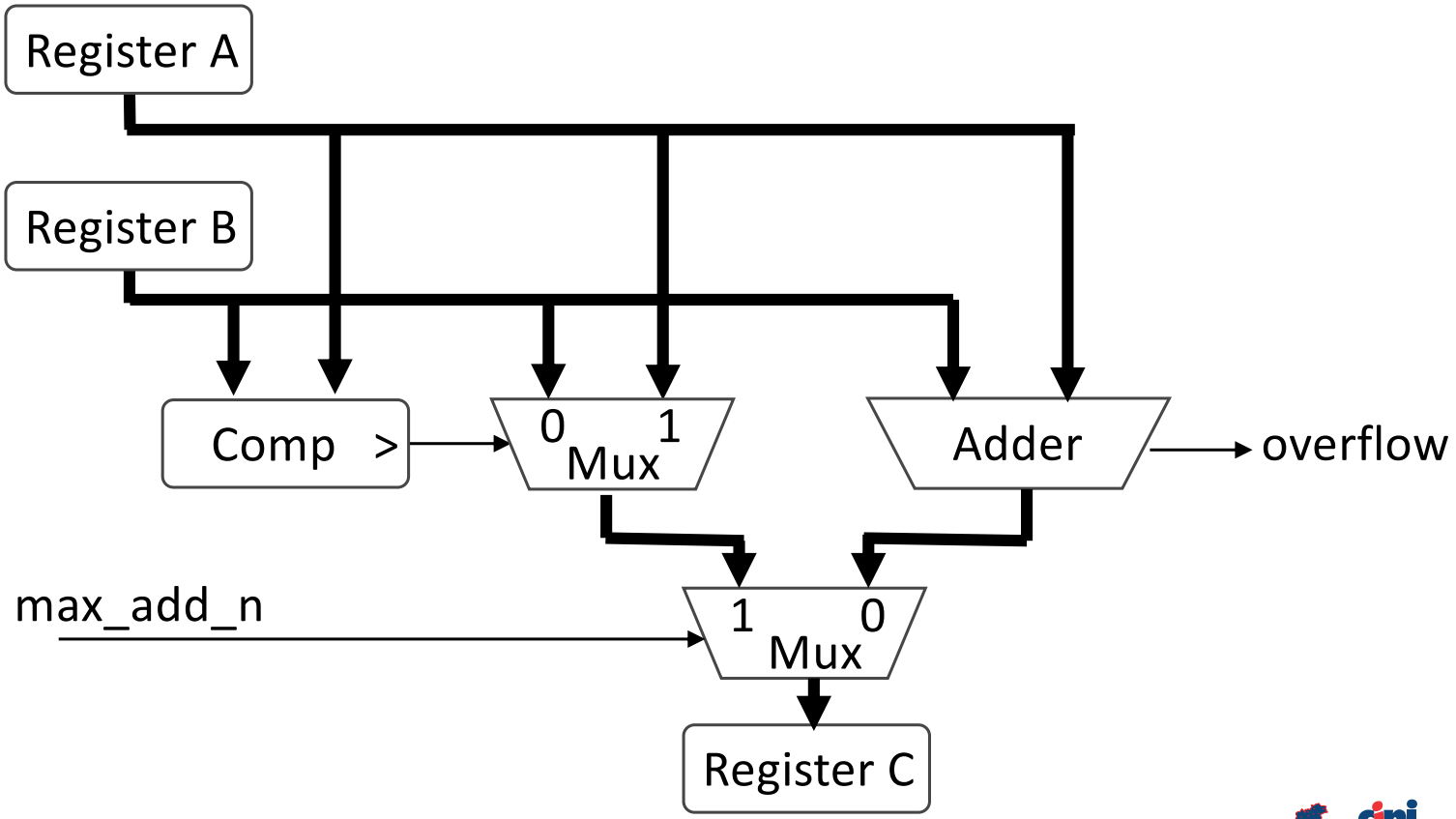


Structural domain – RT level

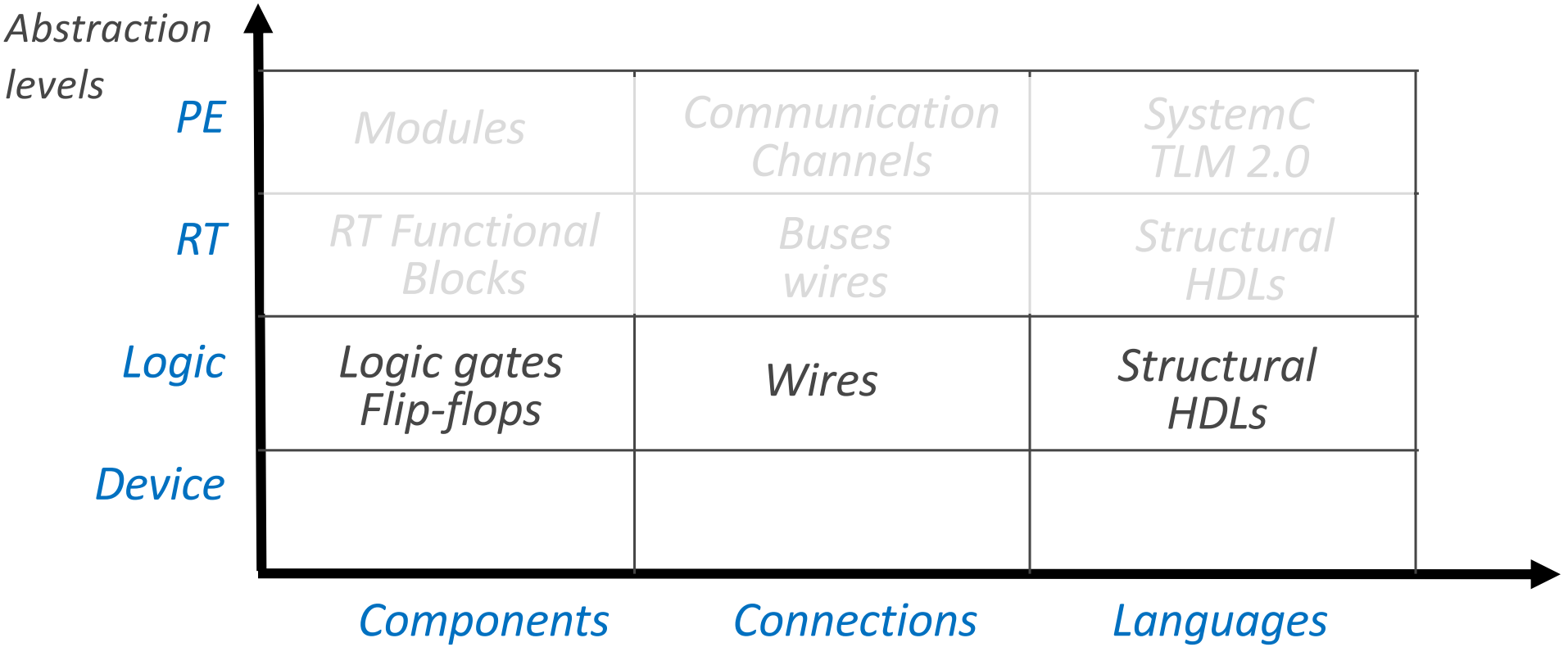


 Rel. 20.02.2020

Example: RT-level structural description



Structural domain – Logic level



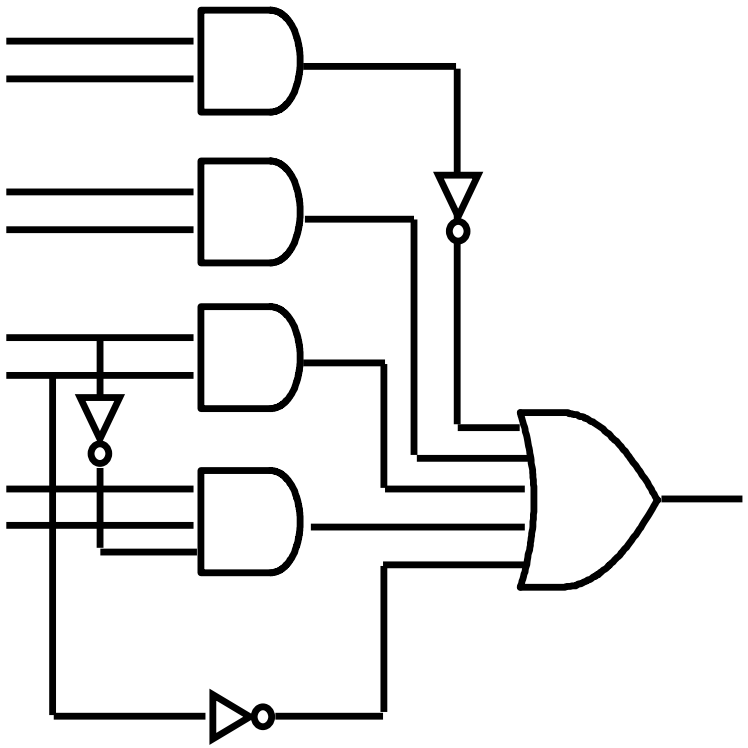
 Rel. 20.02.2020

Netlists

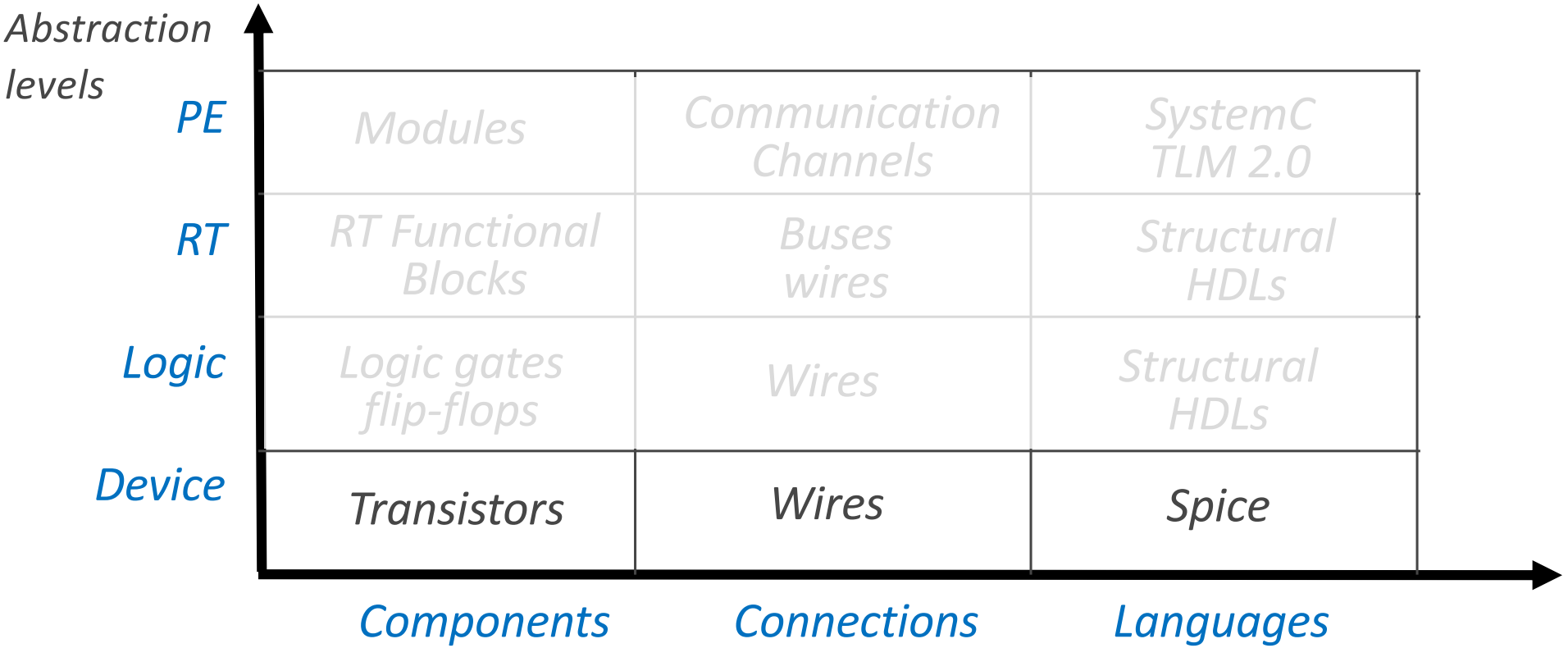
- Structural descriptions at the logic level are usually referred to as *netlist*
- Netlist descriptions are performed according to standards formats, such as: EDIF, JEDEC, etc.

Netlists

- Structural descriptions at the logic level are usually referred to as *netlist*
- Netlist descriptions are performed according to standards formats, such as: EDIF, JEDEC, etc.

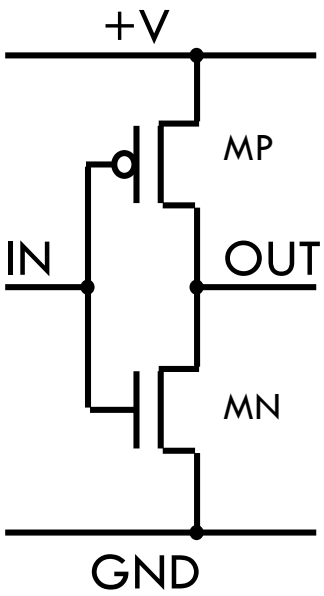


Structural domain – Device level



 Rel. 20.02.2020

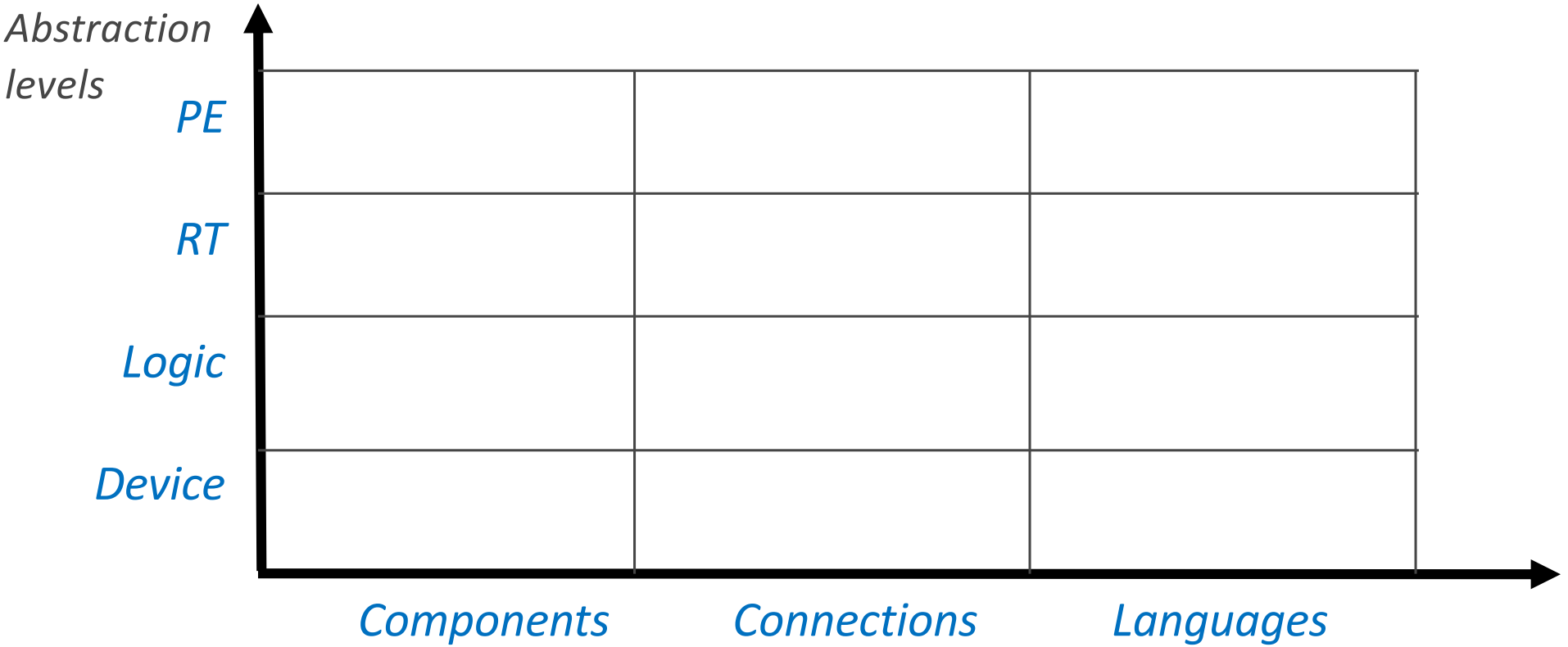
Examples: Transistor level



Outline

- Representation matrix
- Behavioral domain
- Structural domain
- Physical domain
- Design evolution

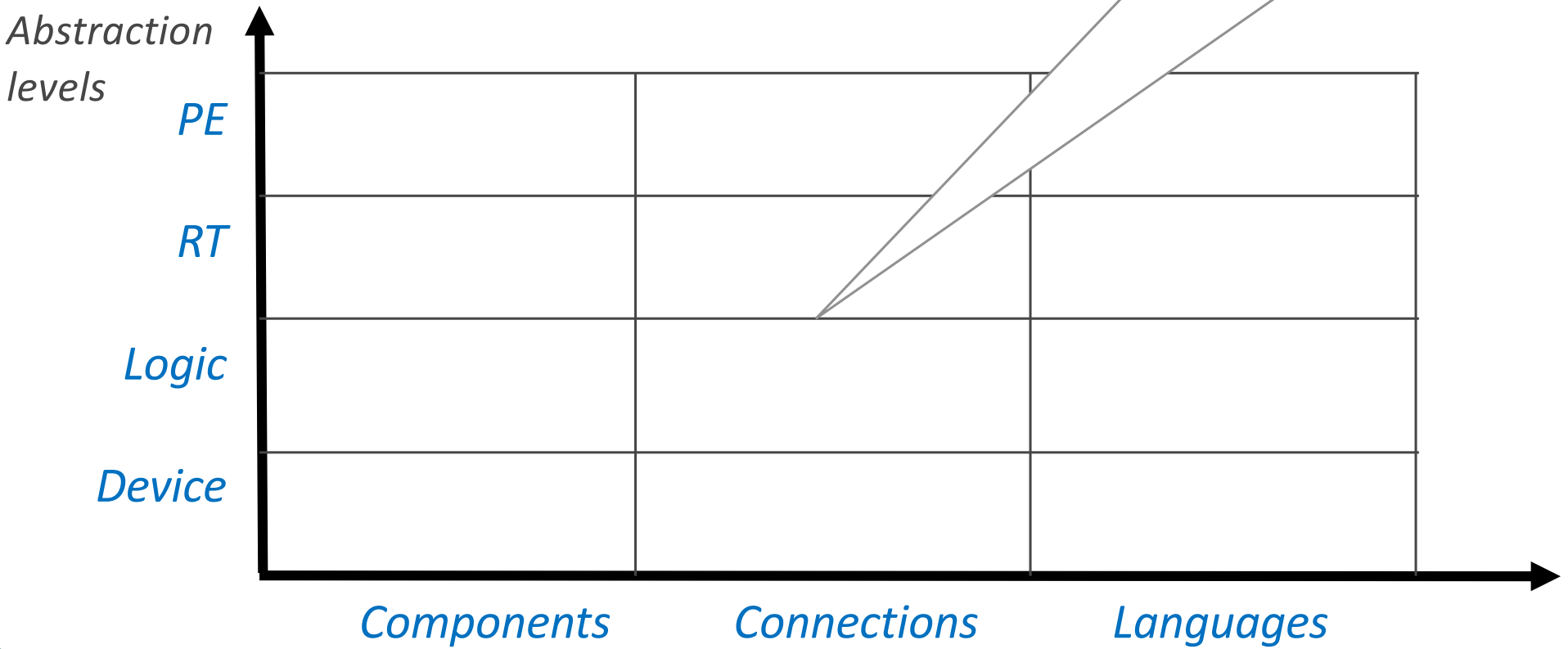
Physical domain



 Rel. 20.02.2020

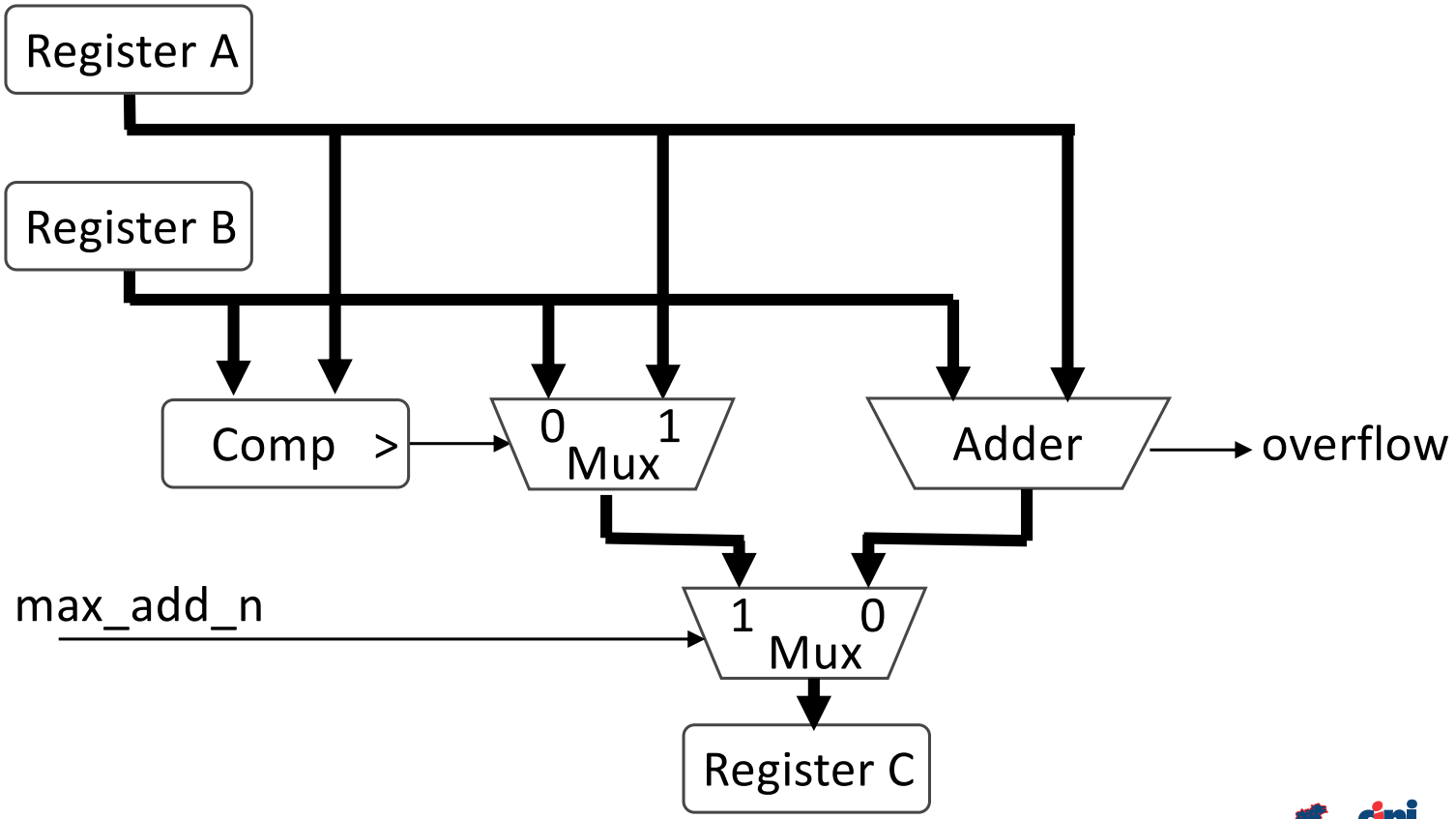
Physical domain

Same as *structural,* but *technology dependent*

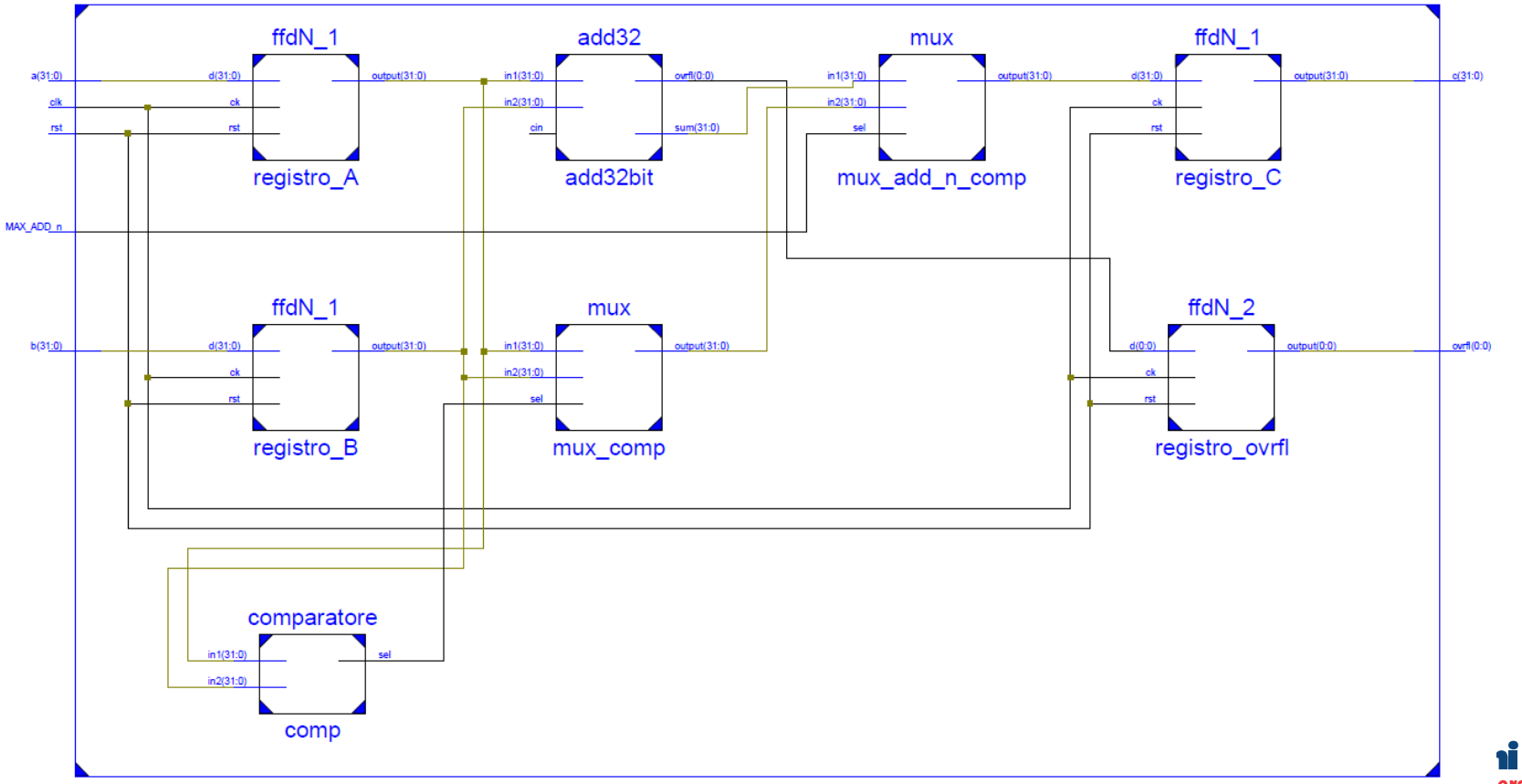


 Rel. 20.02.2020

Example: RT-level structural description

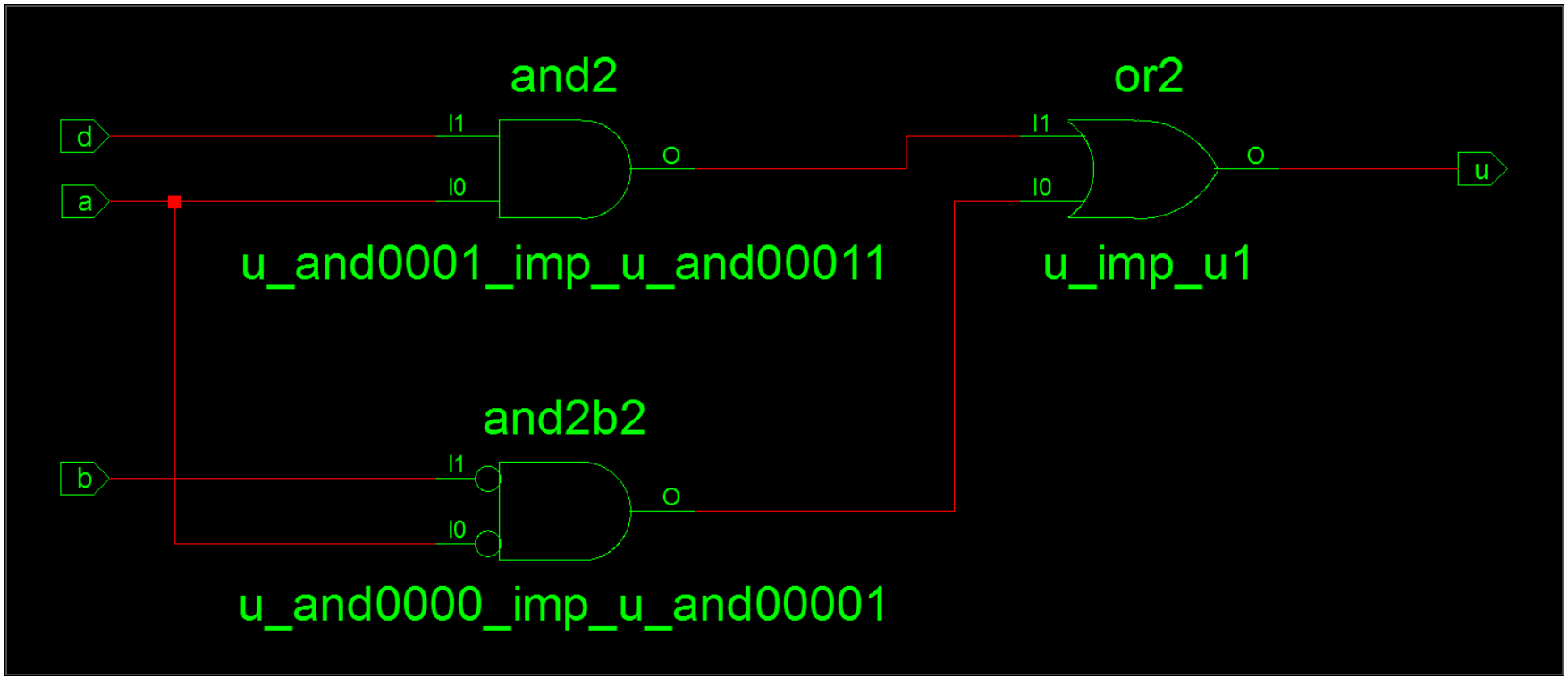


Example: RT-level physical description



ex

Netlist in the physical domain



Outline

- Representation matrix
- Behavioral domain
- Structural domain
- Physical domain
- Design evolution

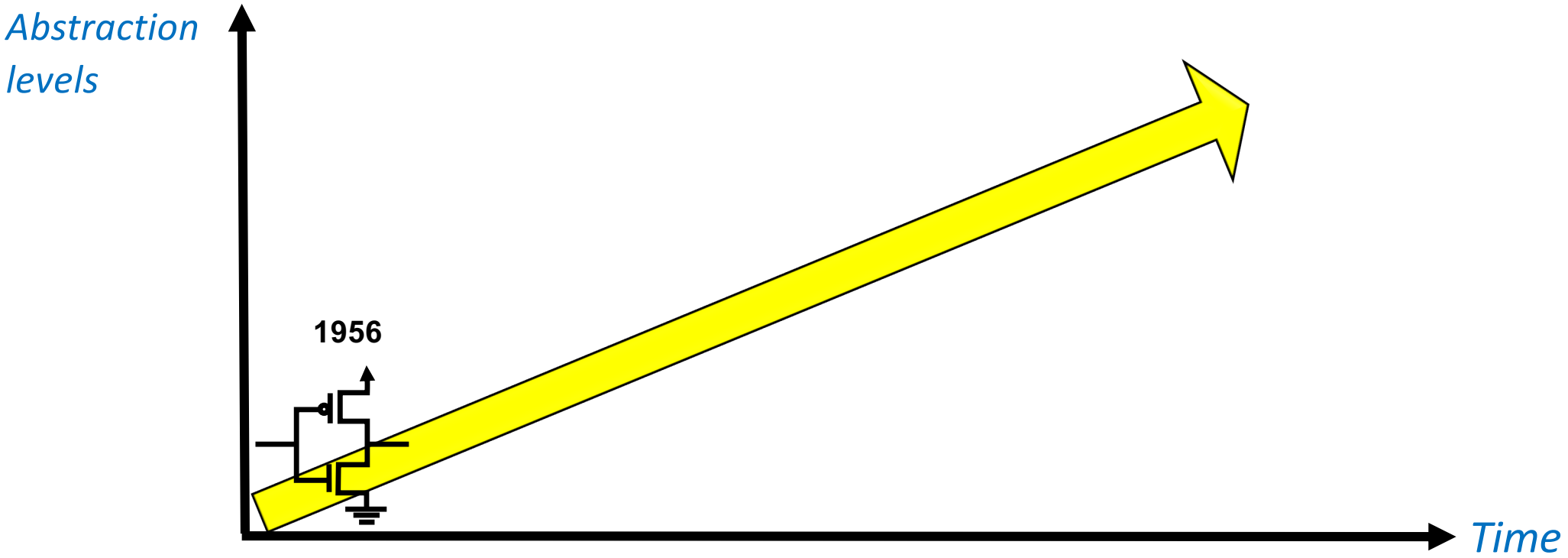
Historical perspective

- To manage complexity, design paradigm has moved to higher abstraction levels
- The history of electronic hardware design has followed a path of increasing levels of abstraction, much like many other technologies.

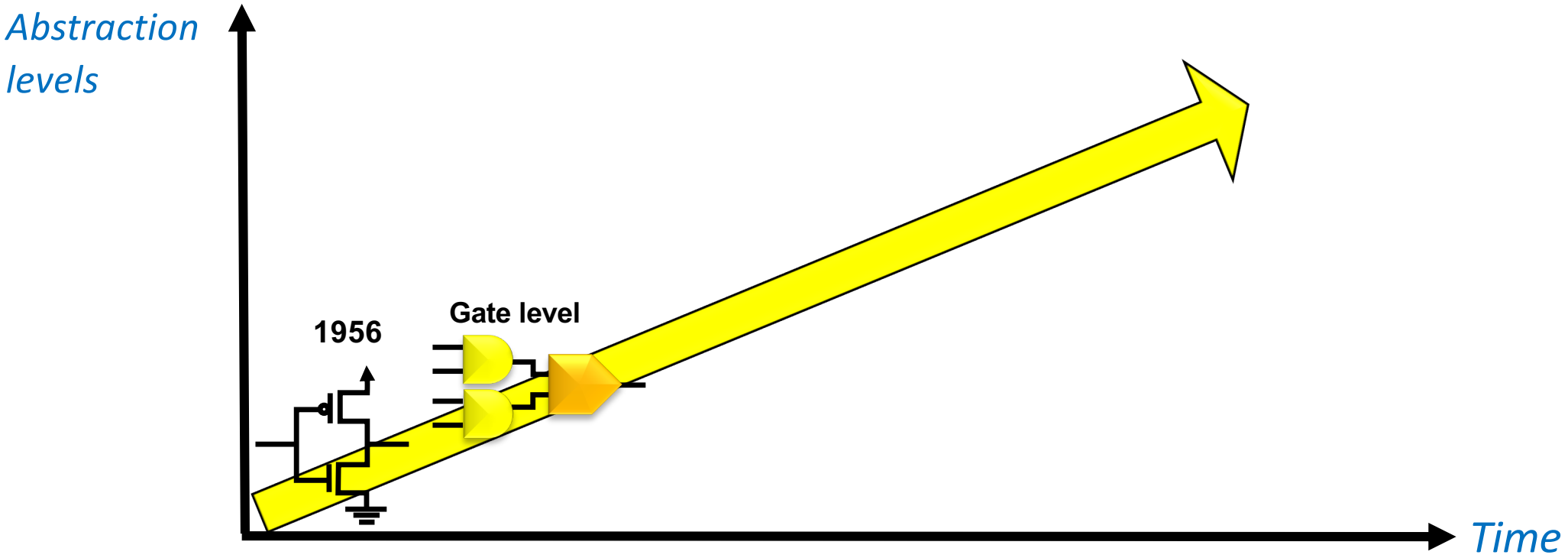
Historical perspective

- As the lower levels of the technology are understood, they become the building blocks of higher levels, allowing designers to abstract details at that level and devote the design effort to more global and presumably important issues.

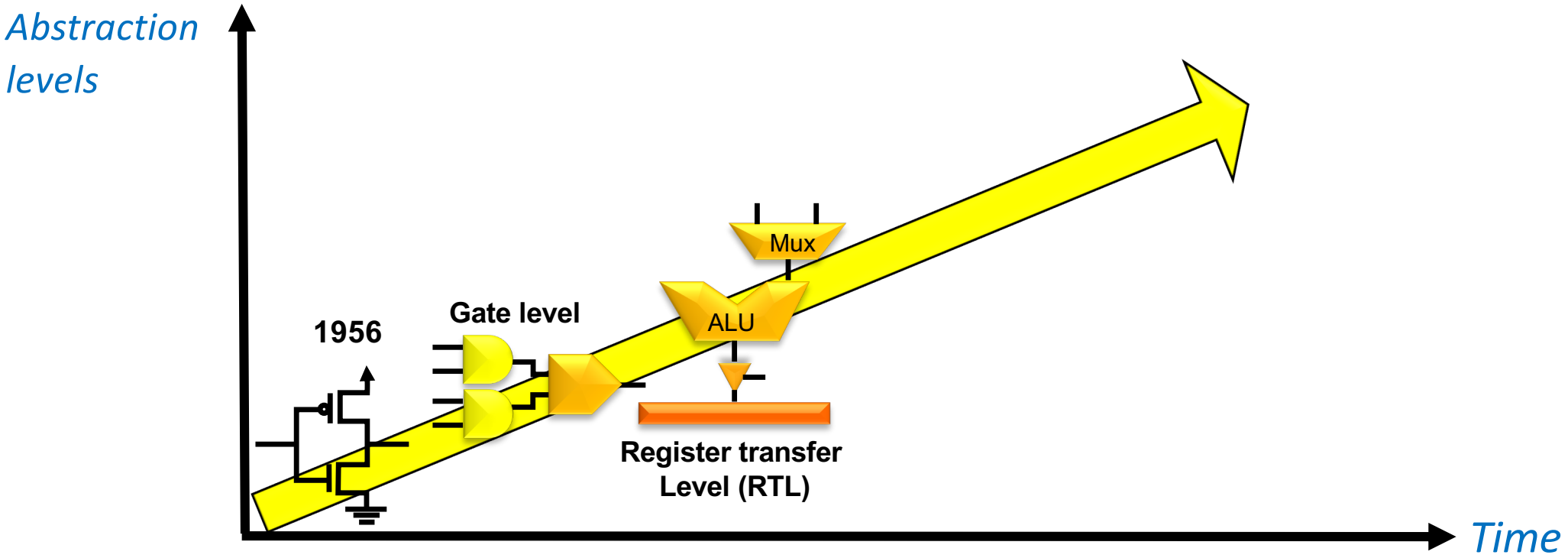
Design Evolution



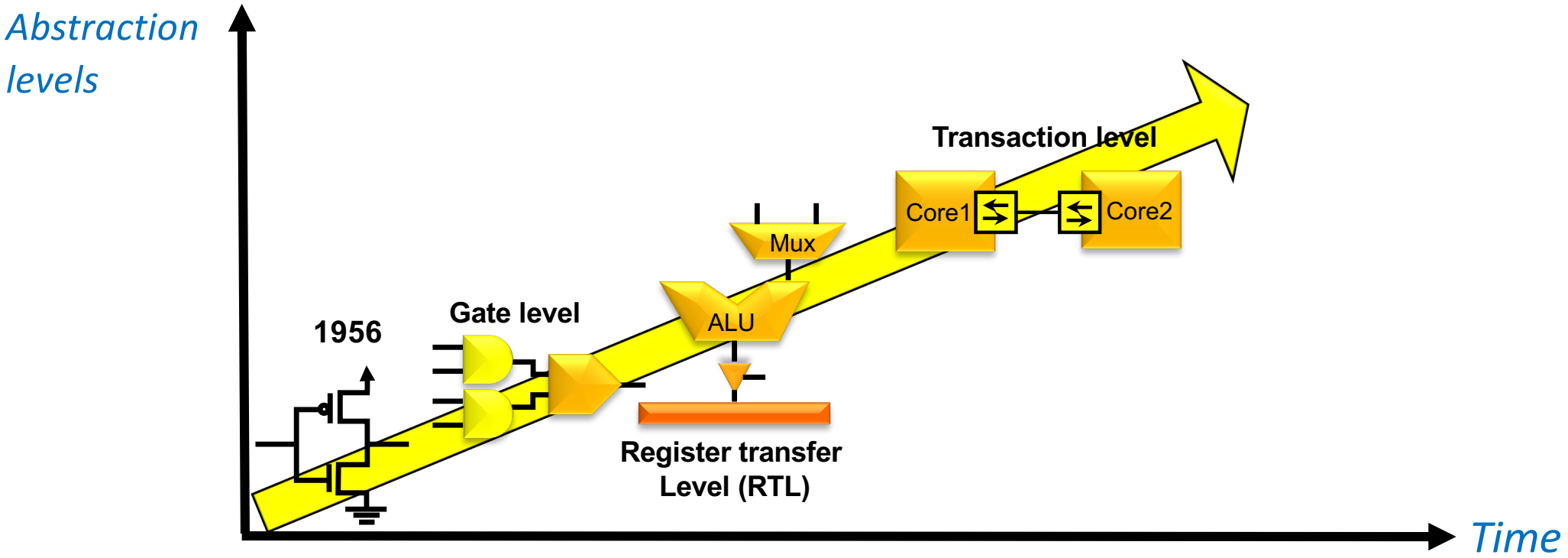
Design Evolution

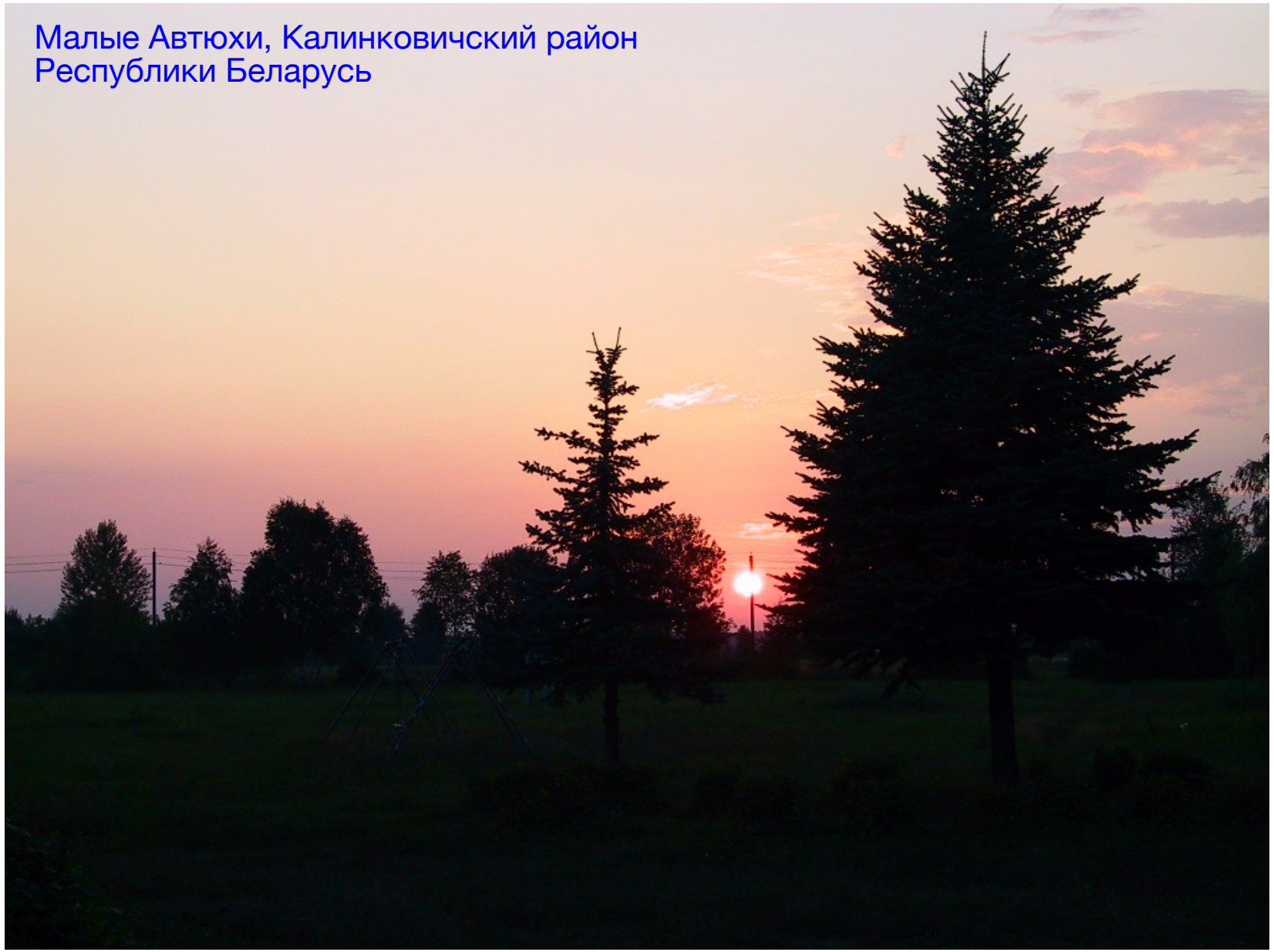


Design Evolution



Design Evolution





Paolo PRINETTO

Director
CINI Cybersecurity
National Laboratory

Paolo.Prinetto@polito.it

Mob. +39 335 227529

CYBER CHALLENGE
CyberChallenge.IT

cini Cybersecurity National Lab

https://cybersecnatlab.it