Professur Technische Informatik
Prof. Dr. Wolfram Hardt

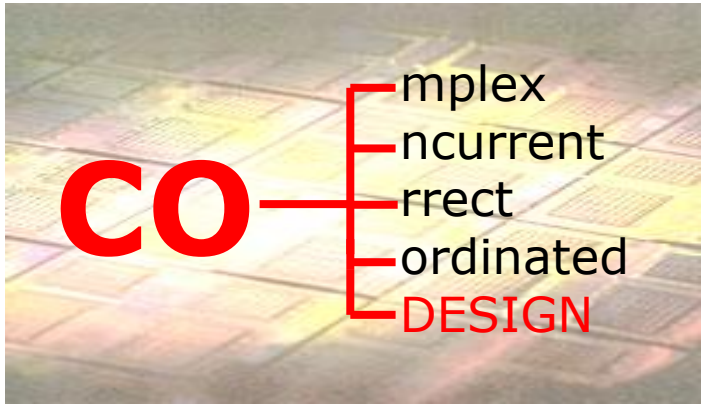
Hardware /Software Codesign I

System Development – Models and Methods

Prof. Dr. Wolfram Hardt
Dipl.-Inf. Michael Nagler

Contents

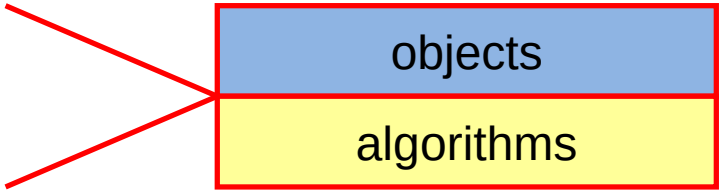
- HW/SW Codesign Process
- Design Abstraction and Views
- Synthesis
- Control/Data-Flow Models
- System Synthesis Models



Designing Embedded Systems

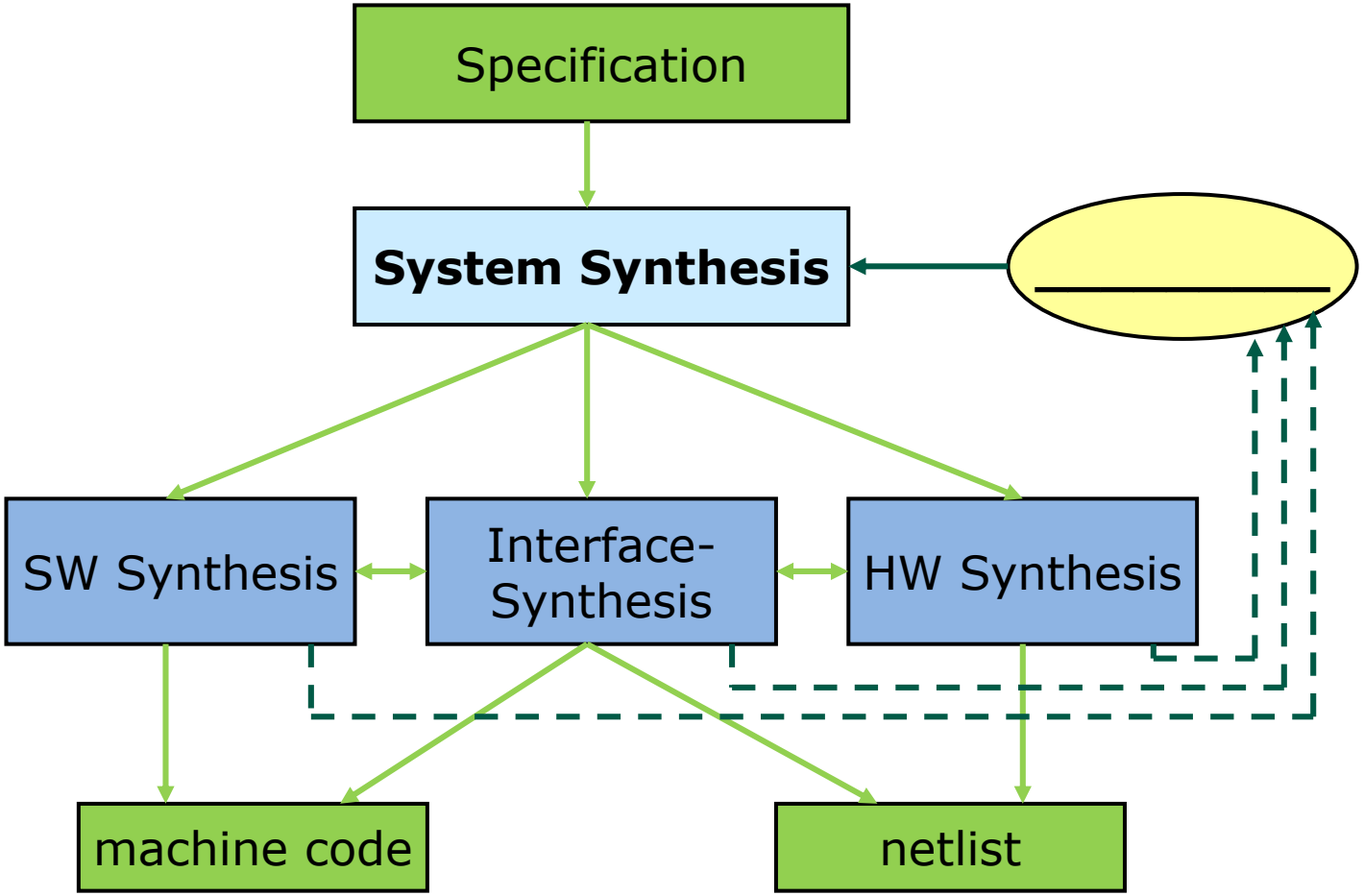
- design structuring
  - representation of requirements
  - necessary for automation of design process

- _____
  - chip - chip
  - board – board
  - hardware – software



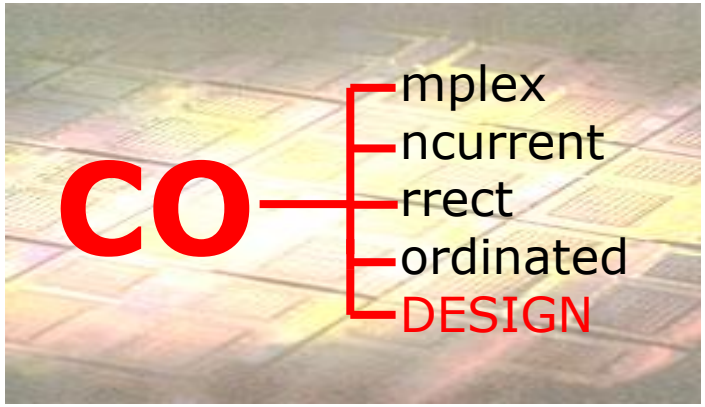
- design steps
  - scheduling
  - allocation
  - binding

HW/SW Codesign Process

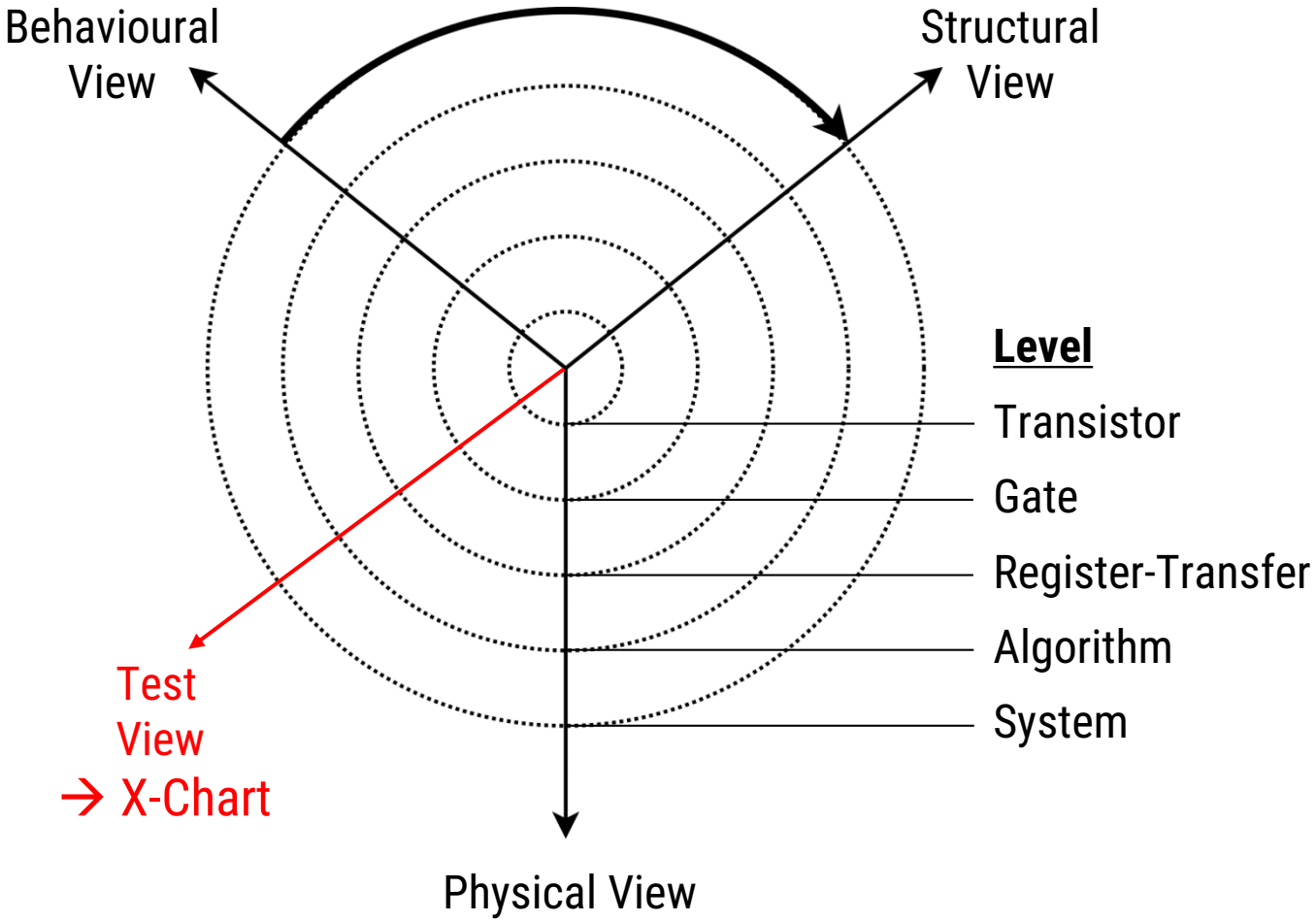


Contents

- HW/SW Codesign Process
- Design Abstraction and Views
- Synthesis
- Control/Data-Flow Models
- System-Synthesis Models



Y-Chart


Behavioural View
Structural View
Level
Transistor
Gate
Register-Transfer
Algorithm
System
Test View
→ X-Chart
Physical View

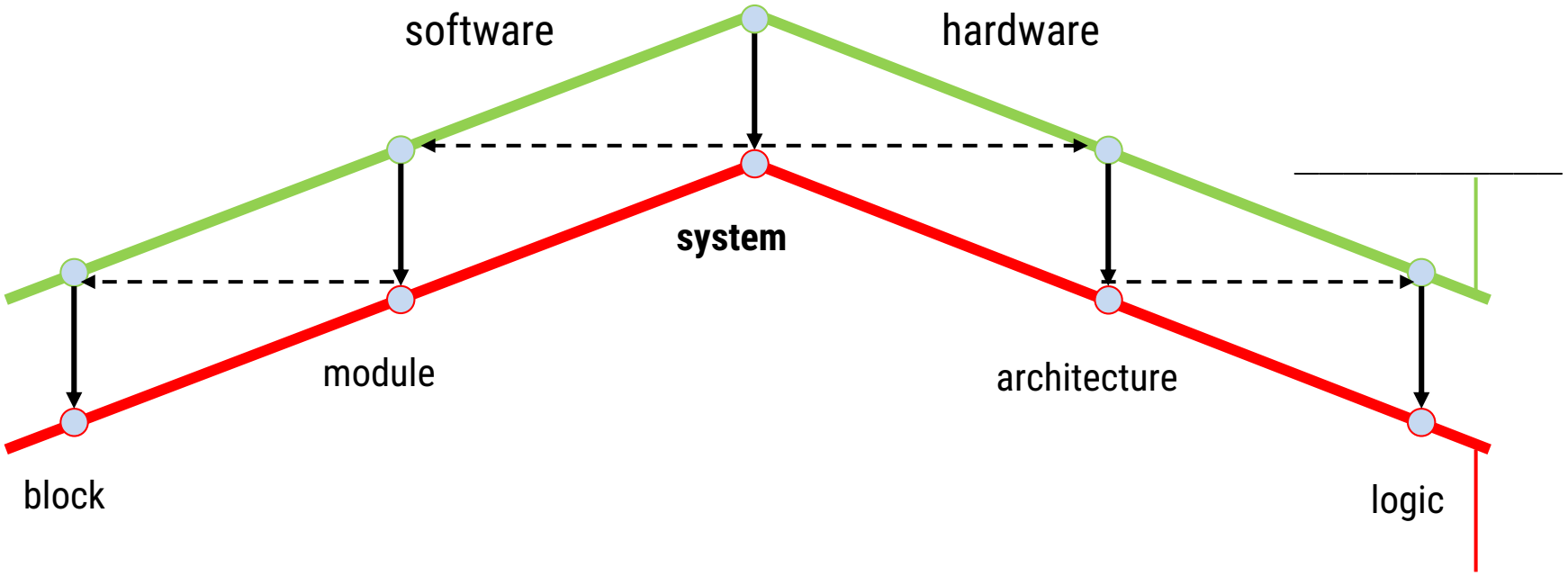
Example

- device: ***MP3 decoder ASIC (HW)***

	Behavioural View	Structural View
system	decode files with 32…256kBit, 32 Bit output quality, …	components for sample detection, configurable decoder, output registers
algorithm	description for each part, e.g. in C / Java	detailed structure for different parts of system
register-transfer	complex modules and their data flow / communication	register-transfer components and connections
gate	_____ _____	_____ _____
transistor	differential equations	transistors, resistors, wires, …

Double-Roof-Model

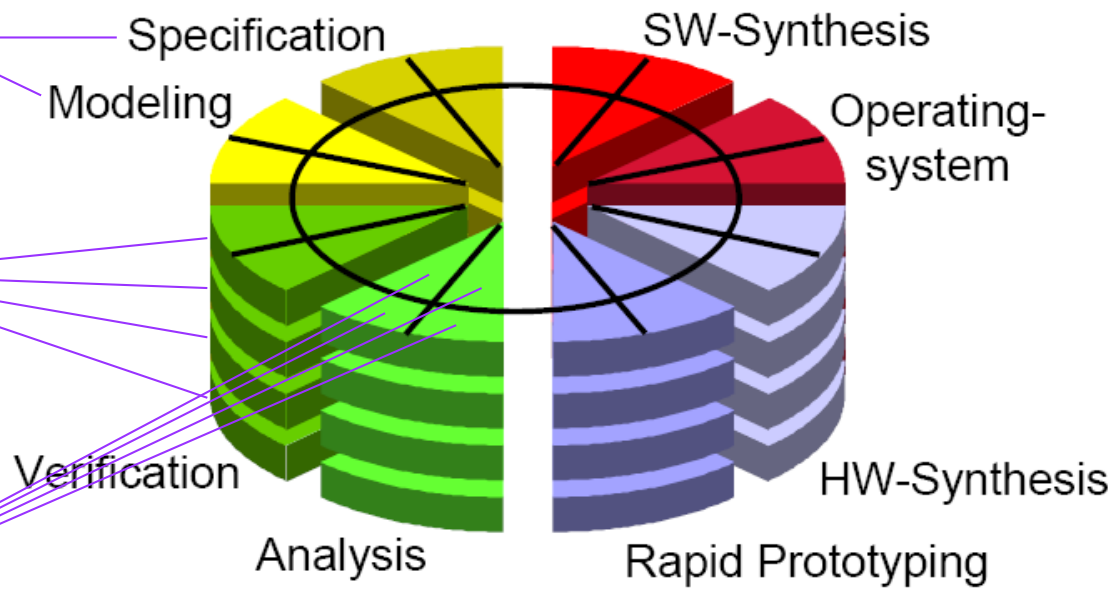
- abstraction of Y/X-chart



→ HW and SW design processes are equally structured

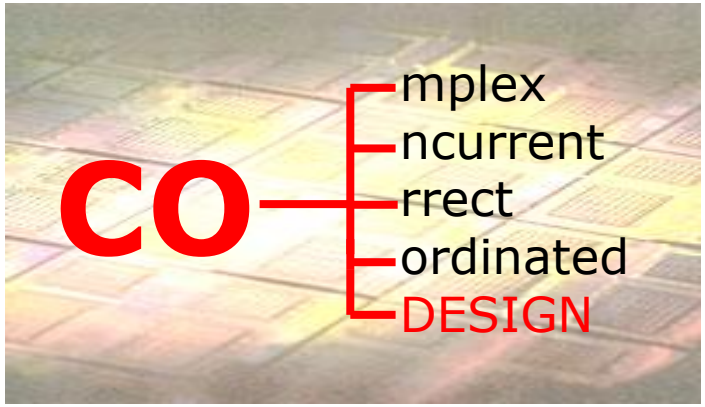
P-Chart

- domain
  - SW-Synthesis
  - …
- level
  - system level
  - algorithm level
  - RT level
  - …
- view
  - behaviour
  - structure
  - test
  - geometry

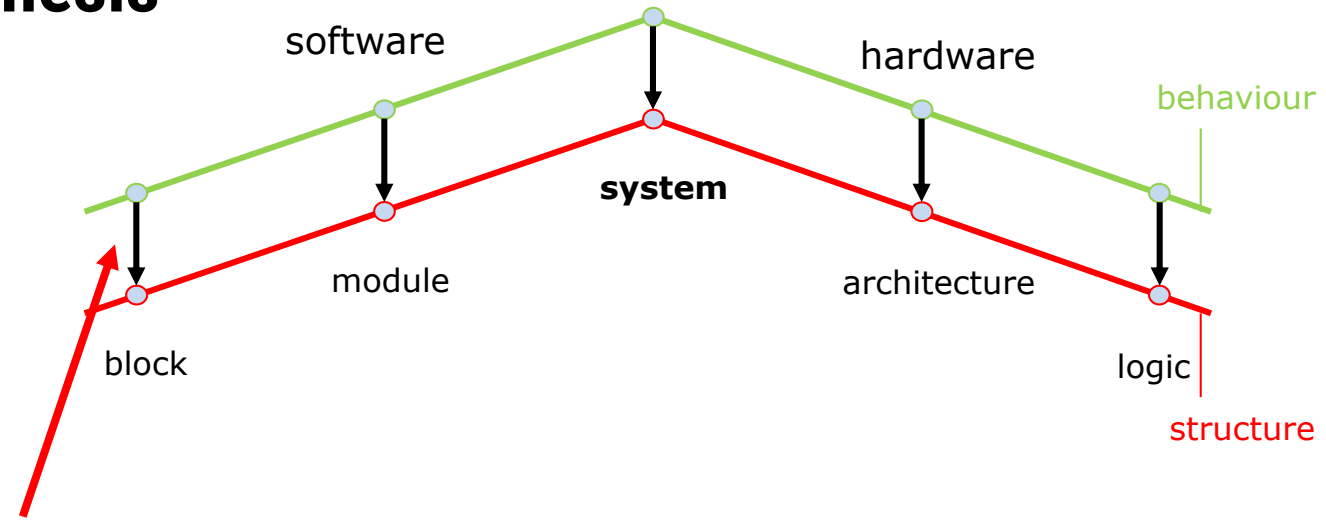


Contents

- HW/SW Codesign Process
- Design Abstraction and Views
- Synthesis
- Control/Data-Flow Models
- System Synthesis Models



Synthesis



- **synthesis =** _____

- tasks for synthesis:
  - allocation: select components
  - binding: map functions to components
  - scheduling: plan execution order

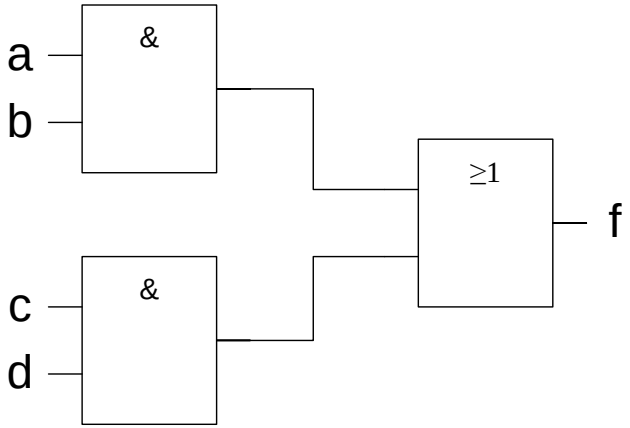
Example

- **behaviour**: Boolean equation

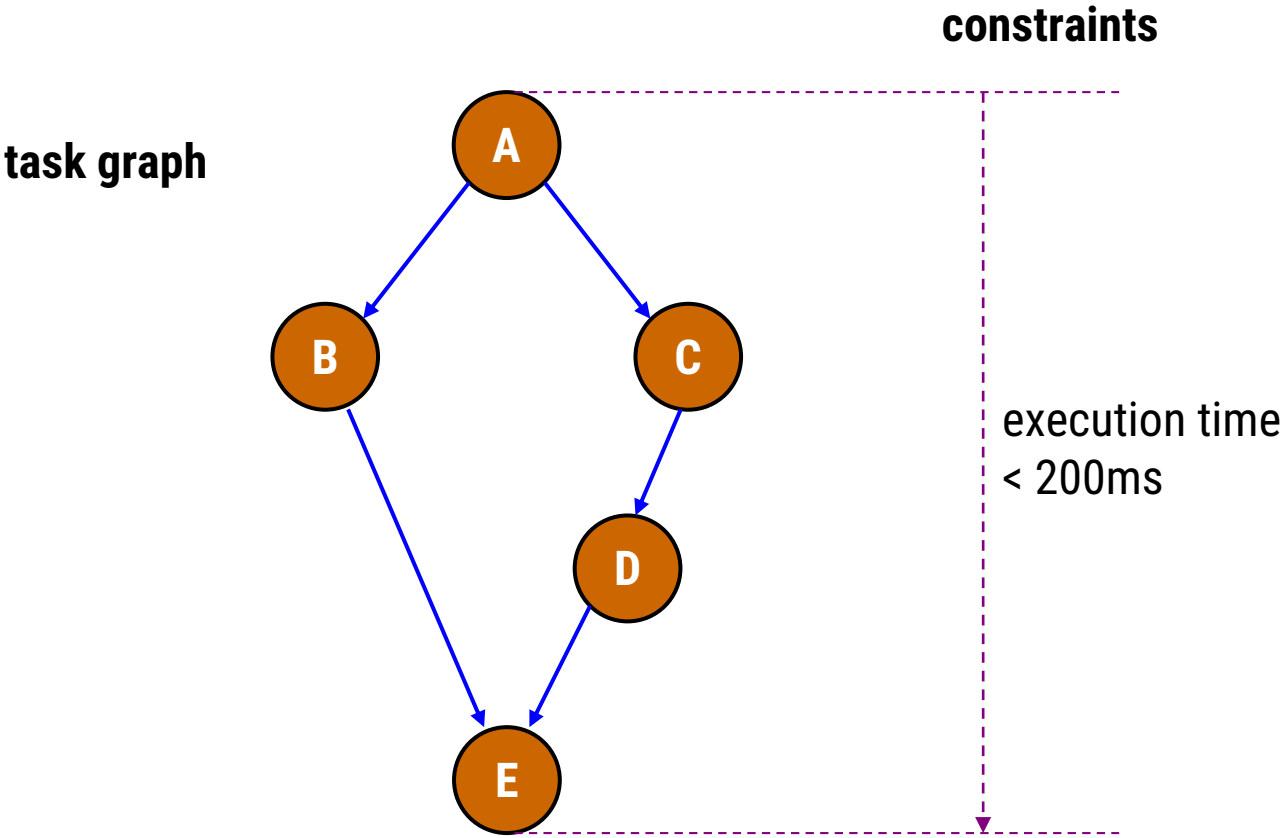
$$f = ab \vee cd$$

synthesis

- **structure**: netlist

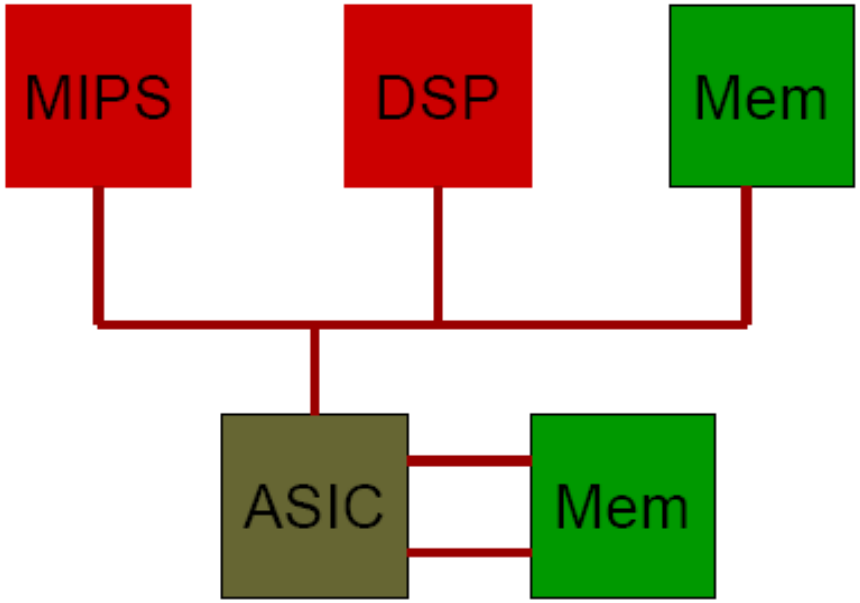


Specification on System Level



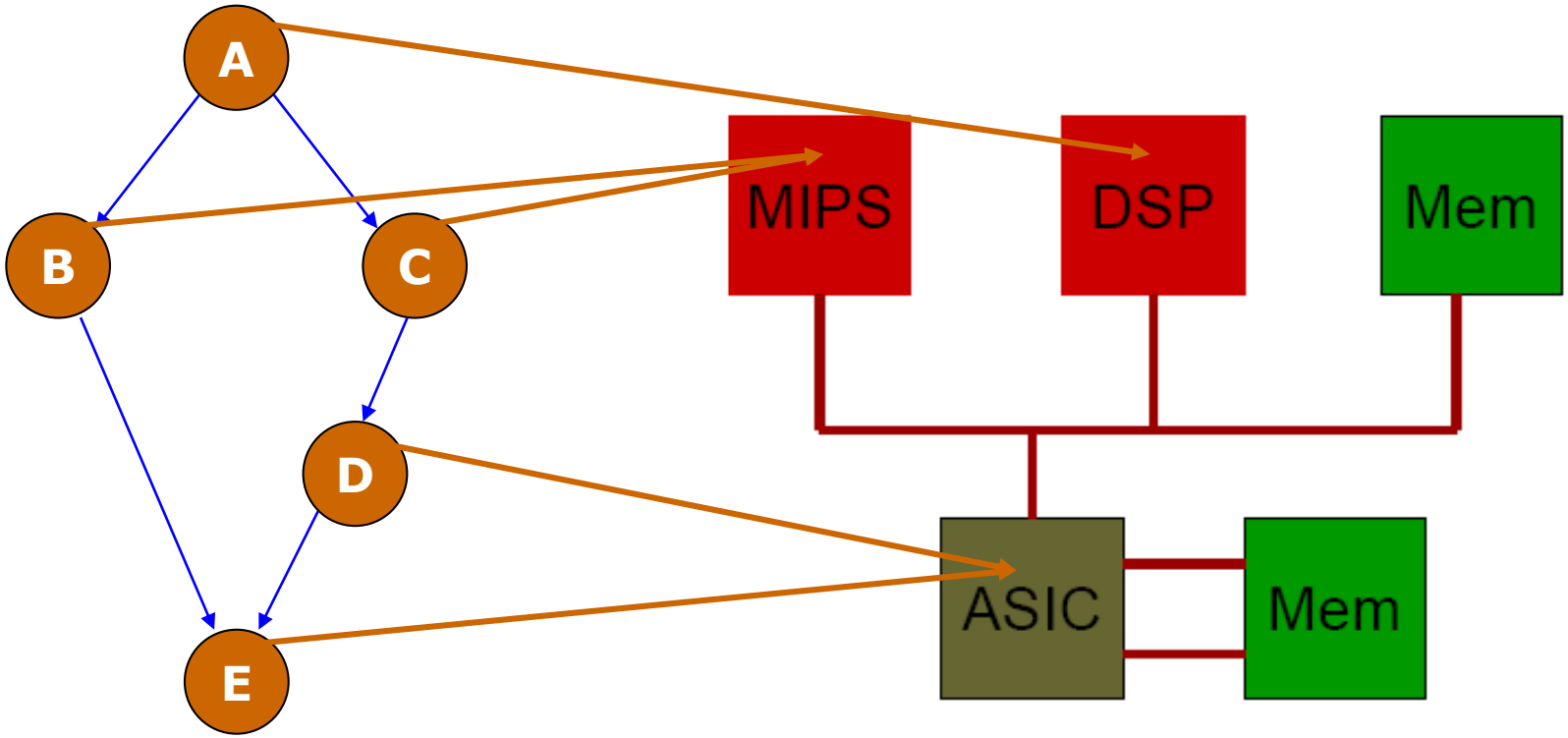
Allocation on System Level

- processor, dedicated hardware
- memory, I/O
- connection structures

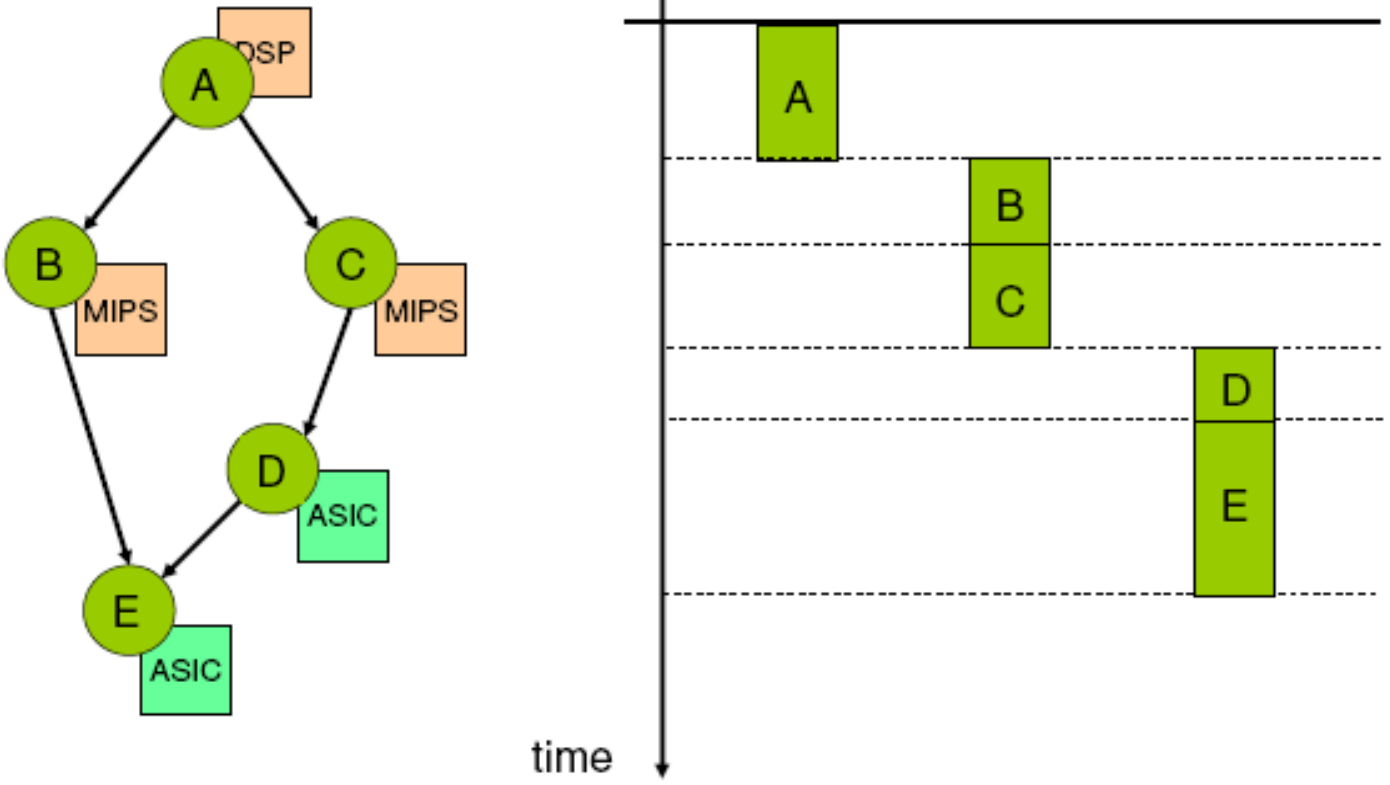


Binding on System Level

- map tasks to resources



Scheduling on System Level



Rating of Design Alternatives

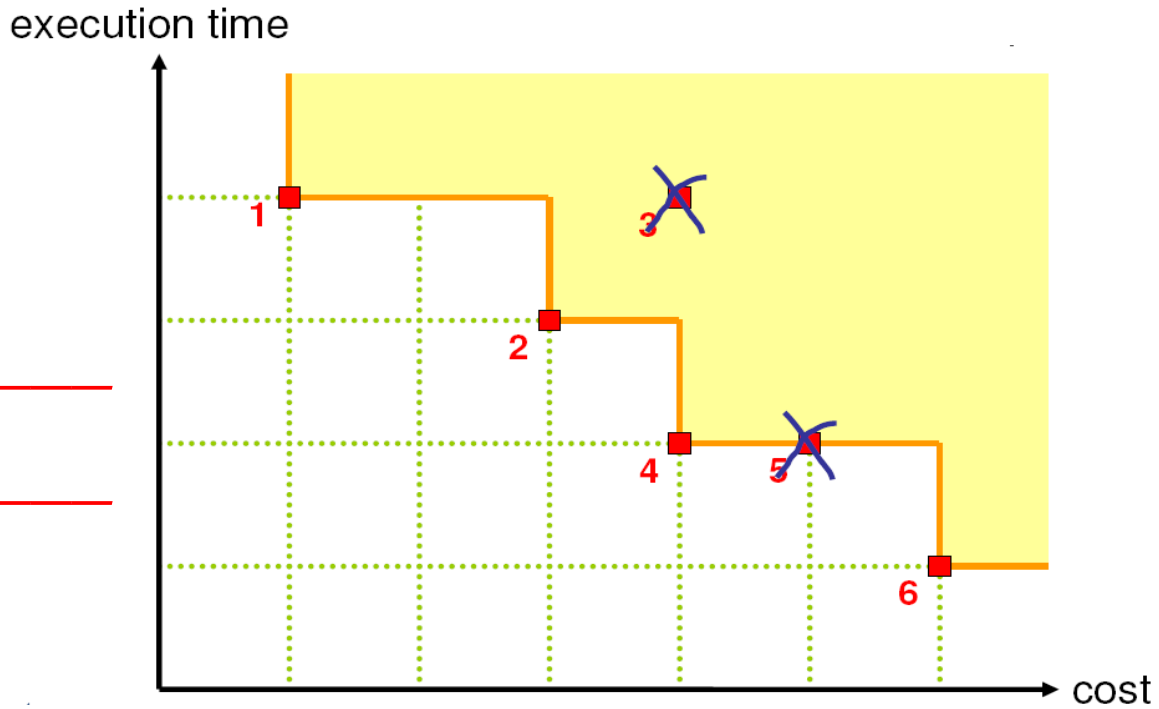
- _____:

  criteria can not be improved without worsen another

→ *1, 2, 4, 6 are*

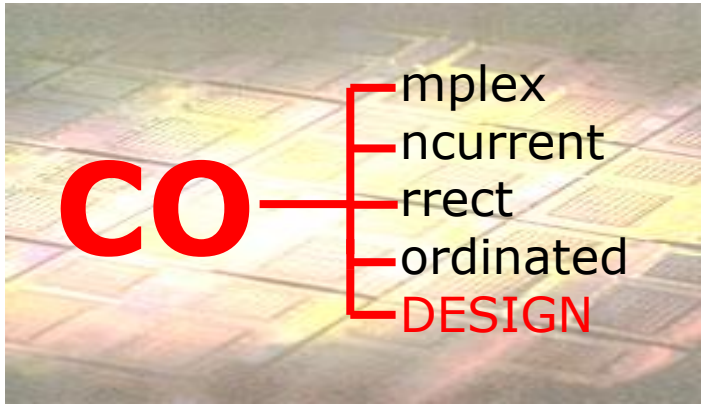
_____

_____



Contents

- HW/SW Codesign Process
- Design Abstraction and Views
- Synthesis
- Control/Data-Flow Models
- System Synthesis Models



Data Structures

- for algorithms (calculation of optimisations) are dedicated and formal models necessary → ***data structures***

- granularity
  - detailing for optimisation parameters
  - abstraction for manageability

- extendibility
  - annotation by optimisation method
  - output generation, if necessary for common optimisation

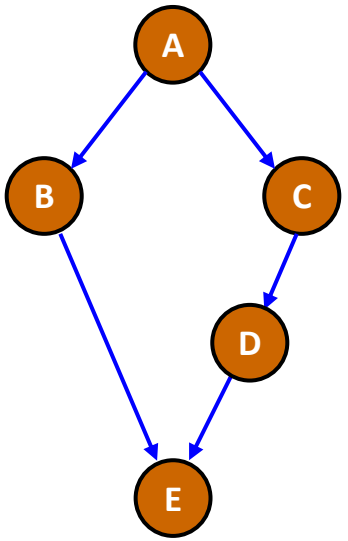
Graph

- modelling by graphs
  - Graph $G = (V,E),\ E \subseteq V \times V$

    set of vertices (nodes) V: *operation, tasks, communication*

    set of arcs (edges) E: *dependencies between nodes*

- example:
  - $G = (V,E)$
  - $V = \{A, B, C, D, E\}$
  - $E = \{(A,B), (A,C), (B,E), (C,D), (D,E)\}$



  - directed graph $(A,B) \neq (B,A)$

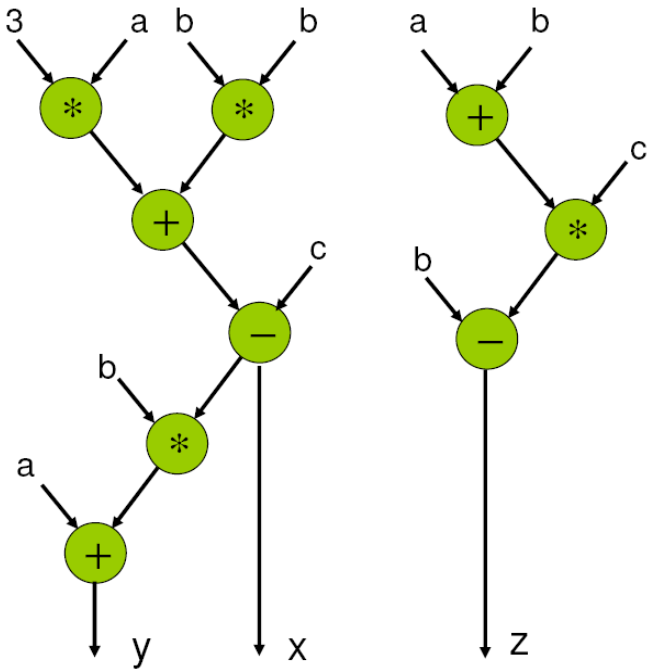
Control and Data Flow Models

- possible dependencies to model by data structure
  - data dependency
  - control dependency
  - _____ (caused by implementation)
- important models
  - data flow graph (DFG)
  - control flow graph (CFG)
  - system modelling graphs (later)

Data Flow Graph (DFG)

- shows data dependencies between calculation or memory units (functions, variables, ALUs, …)
- directed edge, if a producer-consumer relation between two nodes

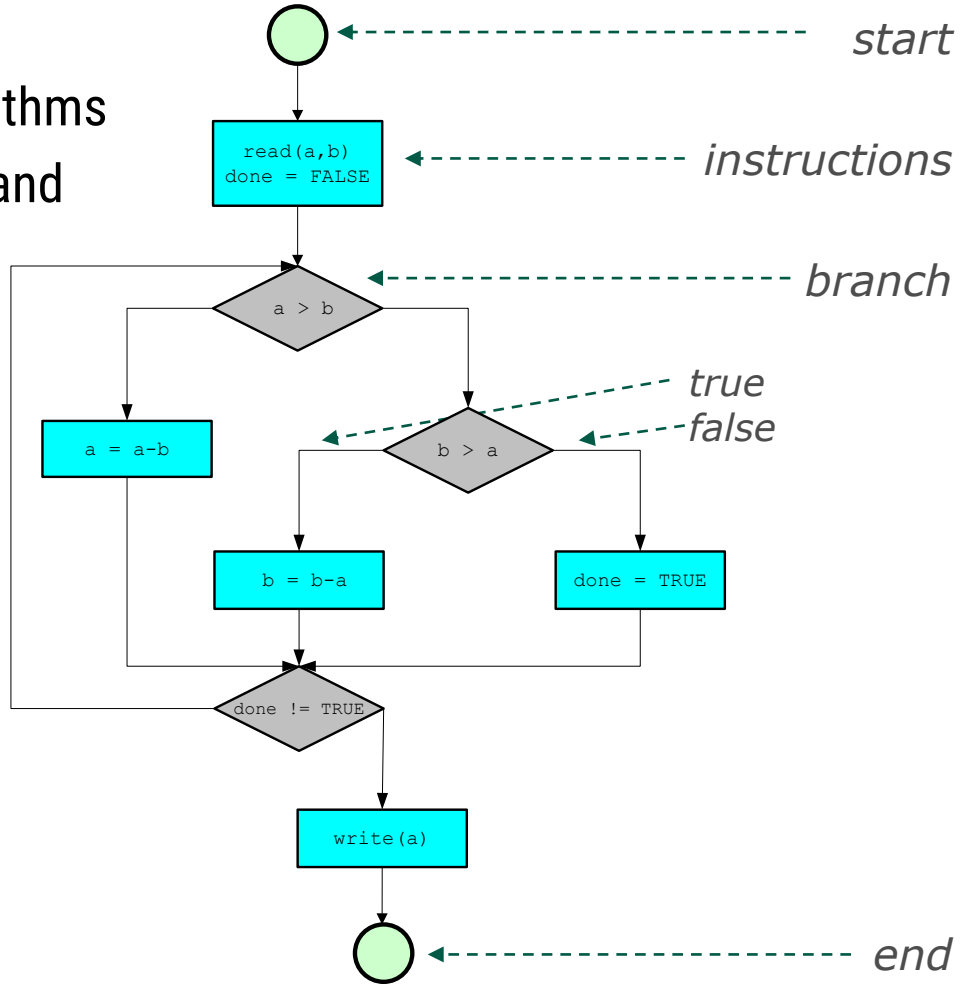
$$x = 3*a + b*b - c;$$
$$y = a + b*x;$$
$$z = b - c*(a + b);$$



Control Flow Graph (CFG)

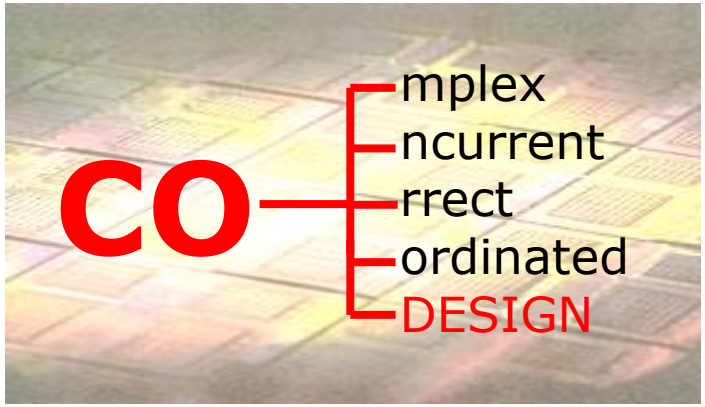
- models control paths of algorithms
- directed graph with one start and one end node

```
foo ()
{   read(a,b);
    done = false;
    repeat
    {    if (a > b)
            a = a - b;
         else
           if (b > a)
              b = b - a;
           else done = true;
     } until done;
     write(a);
}
```



Contents

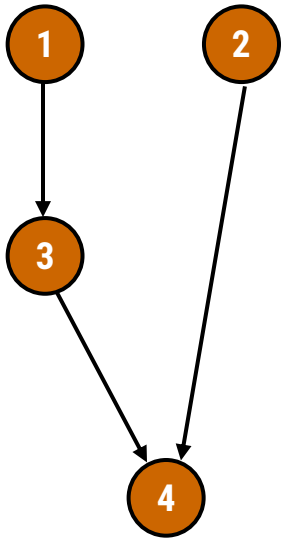
- HW/SW Codesign Process
- Design Abstraction and Views
- Synthesis
- Control/Data-Flow Models
- System Synthesis Models



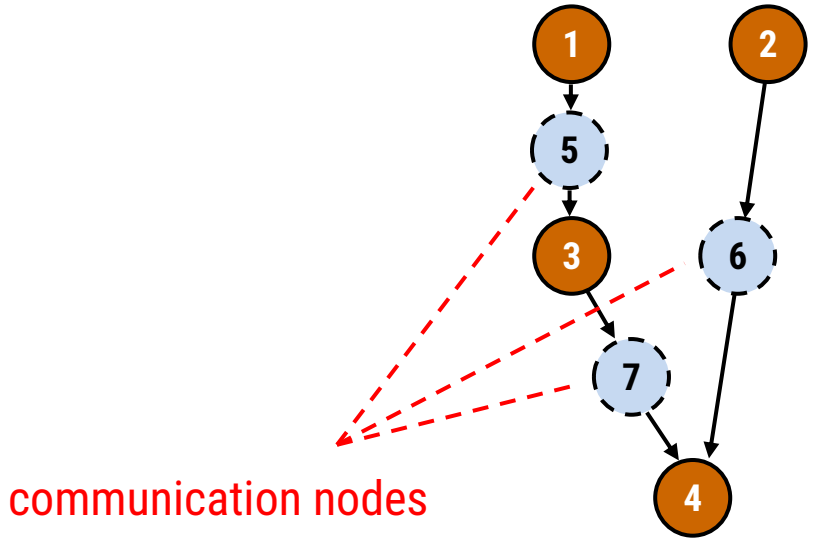
Models for System Synthesis

- problem graph
  - nodes: functional and communication objects
  - edges: dependencies
- architecture graph
  - nodes: functional and communication resources
  - edges: directed communication paths
- **specification graph**
  - **problem graph + architecture graph + _____**

Problem Graph

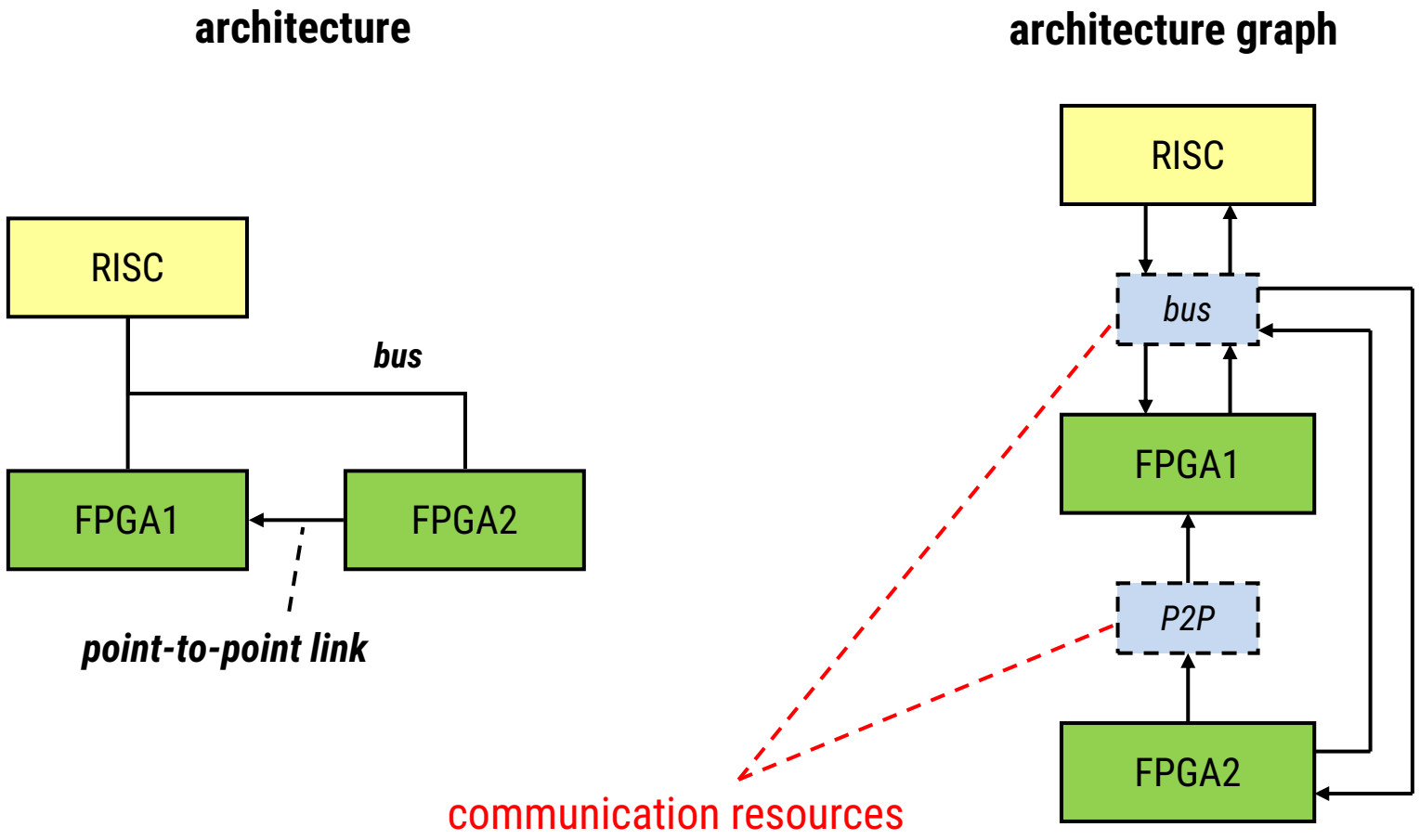


DFG

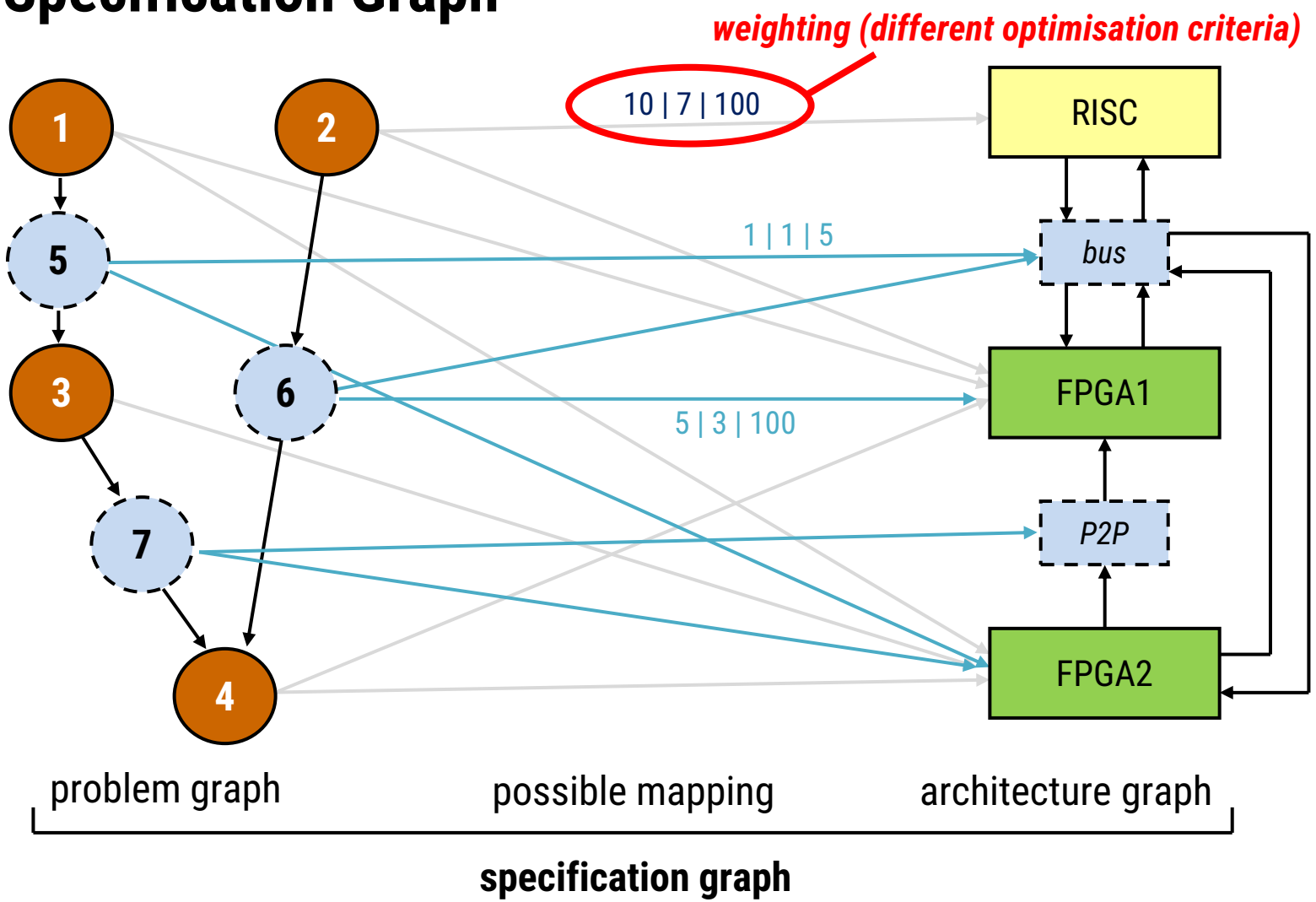


problem graph

Architecture Graph

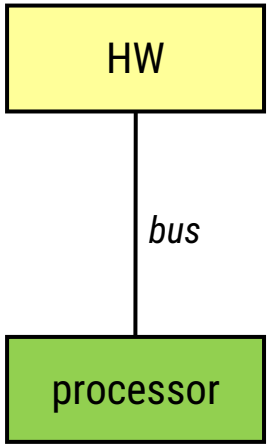


Specification Graph

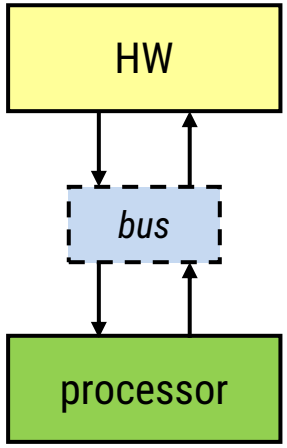


Example

- HW/SW partitioning (_____)



architecture



architecture graph

→ *only two blocks (HW and SW)*