Efficient Correct-by-Construction Design of Avionics Systems

Ingo Sander[1]

Johnny Öberg[1]

Ingemar Söderquist[2]

Gustav Johansson[2]

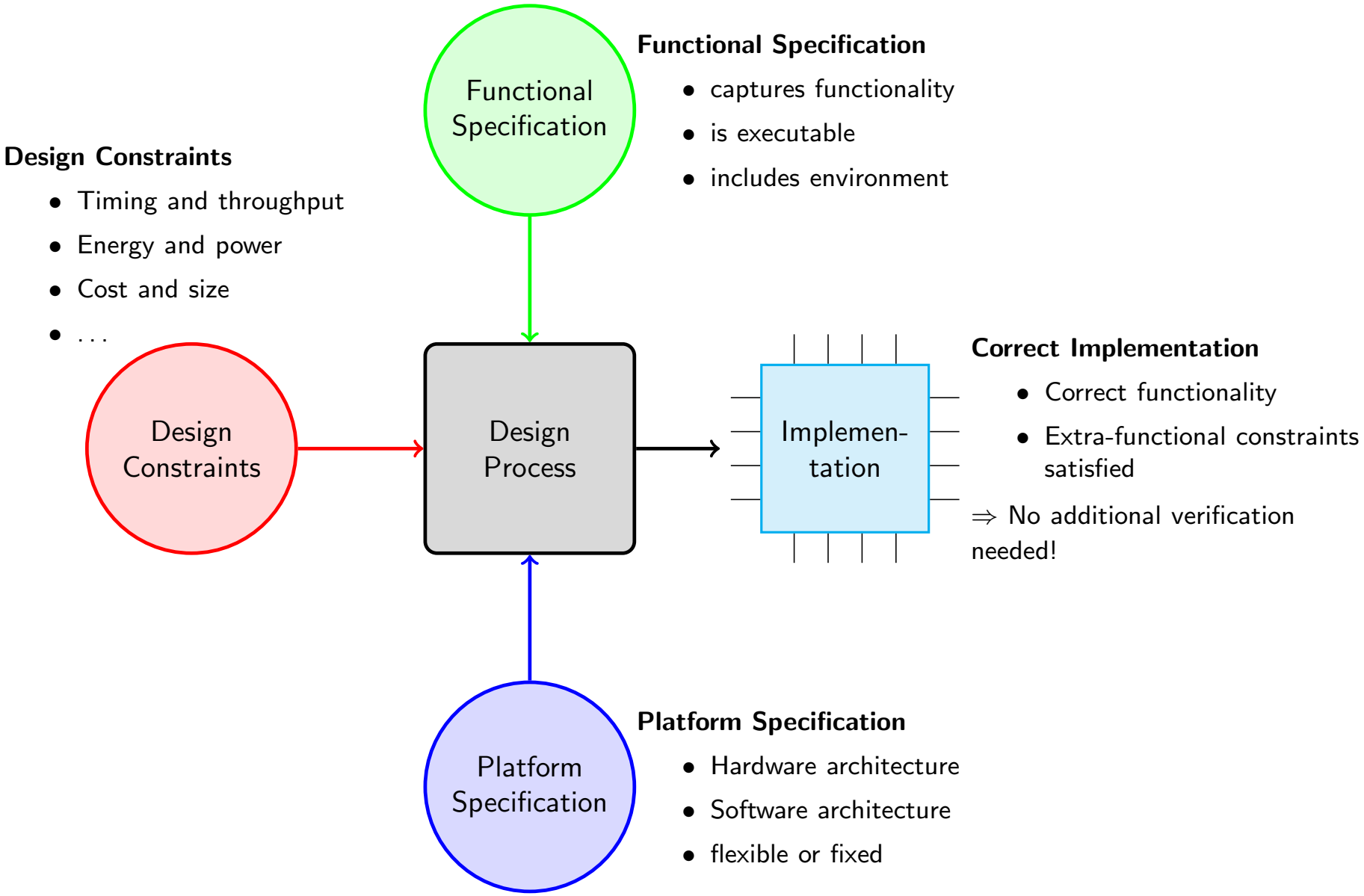
[1]School of Information and Communication Technology
KTH Royal Institute of Technology, Stockholm, Sweden

[2]Saab AB, Linköping, Sweden

Aerospace Technology Congress 2016

SAAB

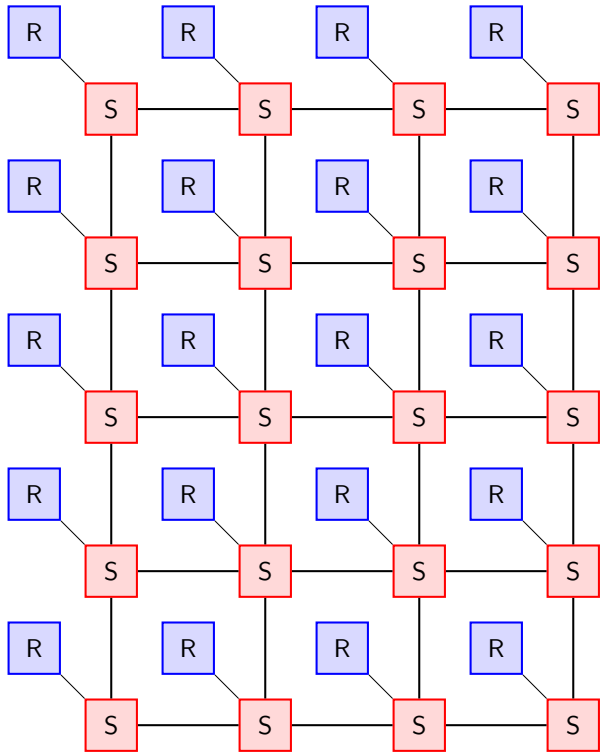
Correct-by-Construction Design: The Dream



Design Challenge

Real-Time in the Many-Core Era

- Technology advances lead to
  - increasingly parallel, powerful and complex architectures
  - increasingly advanced and demanding applications
- Difficult to verify realtime requirements

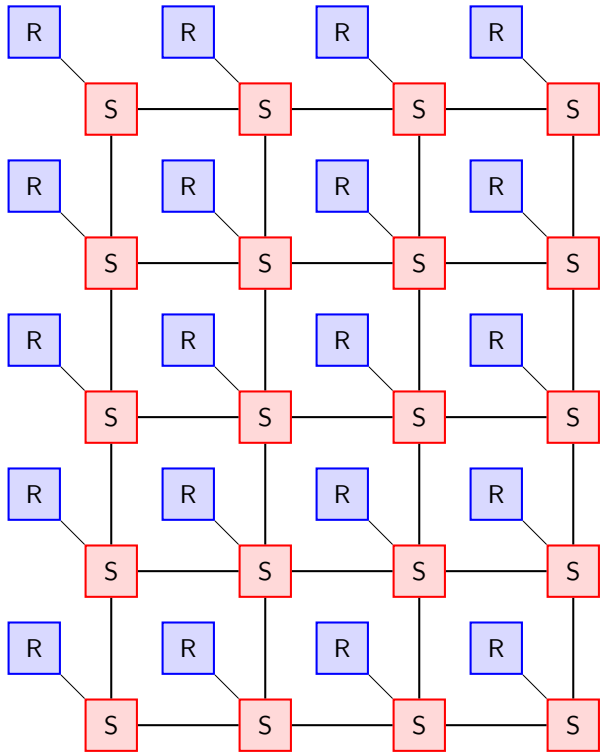


Network-on-Chip

Design Challenge

Real-Time in the Many-Core Era

- Technology advances lead to
  - increasingly parallel, powerful and complex architectures
  - increasingly advanced and demanding applications
- Difficult to verify realtime requirements



Network-on-Chip

We have already problems with complexity today! How do we design tomorrow's "sea-of-cores" real-time systems?

Embedded Real-Time Software Design: Current Situation

Very difficult to accurately estimate real-time performance

- Huge difference between average and worst-case execution time

Consequence

- New designs are rather based on old experience than on performance analysis
- Large safety-margins in form of more powerful components and extra communication bandwidth
- Verification costs are exploding

Embedded Real-Time Software Design: Current Situation

Very difficult to accurately estimate real-time performance

- Huge difference between average and worst-case execution time

Consequence

- New designs are rather based on old experience than on performance analysis
- Large safety-margins in form of more powerful components and extra communication bandwidth
- Verification costs are exploding

Current Situation

Far away from correct-by-construction!

Why is it so difficult to estimate software performance?

- Computation time is difficult to predict in advanced commercial processors due to mechanisms aimed to improve average case performance, e.g. caches, advanced pipelines
- Communication time is difficult to predict im multiprocessors due to
  - communication via shared resources, e.g. memory, communication channel
  - location dependent communication time (NUMA-architectures, NoC)
  - asynchronous communication mechanisms

Why can we accurately predict hardware performance?

- Communication: Elegant synchronisation mechanism: hardware clock
  - Well-defined order of events $\Rightarrow$ predictable communication
  - Longest computation path determines clock period
- Computation: Foundation for digital hardware is simple
  - Mathematical foundation: Boolean algebra
  - Small set of basic blocks: All logic can be realized with NAND gates
  - Accurate timing information for hardware components
- Hardware design languages reflect the underlying parallel architecture and synchronous communication
- Synthesis tools can accurately predict timing properties of implementation

Why can we accurately predict hardware performance?

- Communication: Elegant synchronisation mechanism: hardware clock
  - Well-defined order of events $\Rightarrow$ predictable communication
  - Longest computation path determines clock period
- Computation: Foundation for digital hardware is simple
  - Mathematical foundation: Boolean algebra
  - Small set of basic blocks: All logic can be realized with NAND gates
  - Accurate timing information for hardware components
- Hardware design languages reflect the underlying parallel architecture and synchronous communication
- Synthesis tools can accurately predict timing properties of implementation

Current situation

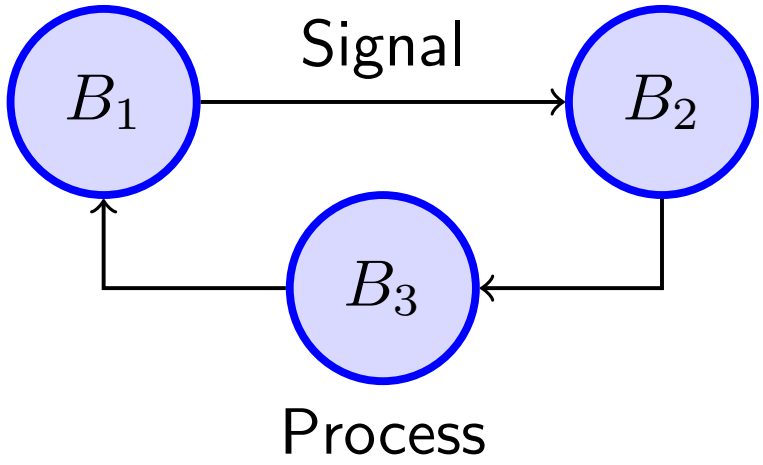
- In practice: Correct-by-Construction
- Simple platforms with formal foundation enable powerful tools

What is needed in Software Design?

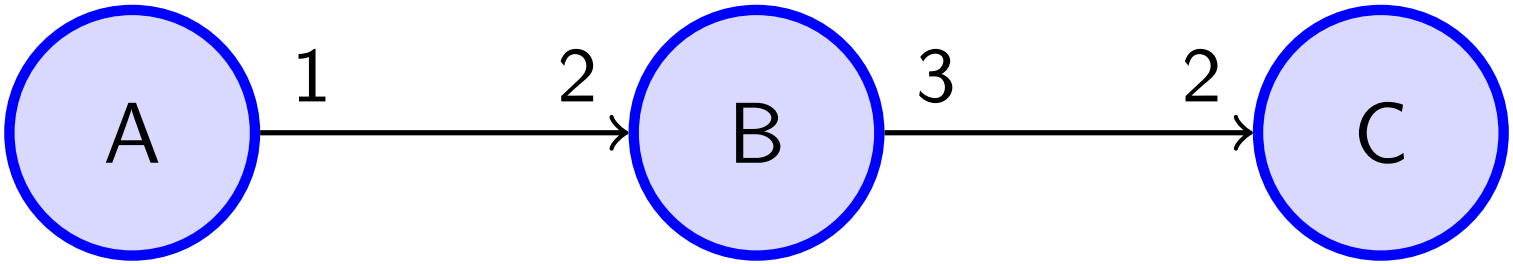
- Formally analysable application models based on a formal foundation to enable formal analysis and reasoning
- Predictable platforms providing service guarantees
- Analysis methods enabling powerful design tools
- Design entry language based on the formal foundation
  - Simulation of the application model
  - Automatic abstraction of formal analysable model

Models of Computation (MoCs)

- specifies semantics of computation and comunication
- abstracts from design languages
- enables formal analysis
  - performance analysis and design space exploration
  - formal verification
  - hardware and software synthesis
- constitute active research area
  - continuous development of new techniques and tools

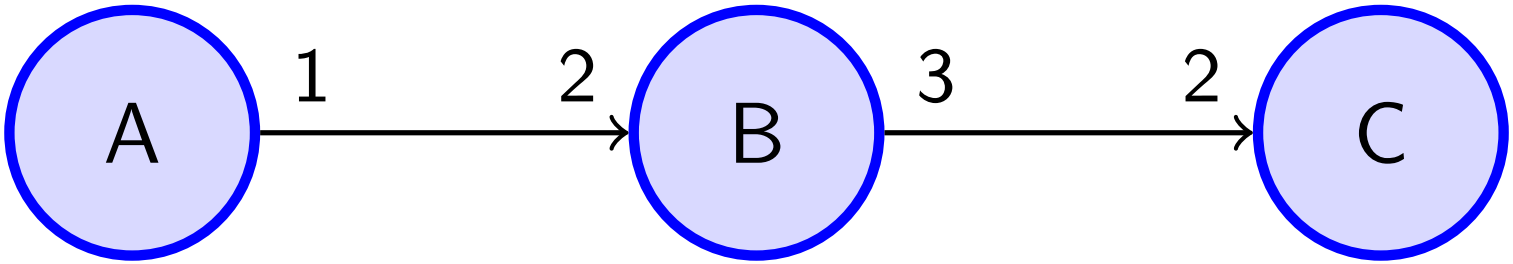


Motivational Example: Streaming Media Application



- If designer is not aware of MoCs, similar application will likely be implemented with
  - RTOS to schedule the processes
  - message queues for process communication
- . . . but this comes with an overhead in form of
  - the RTOS itself
  - safety margin for message queue buffers

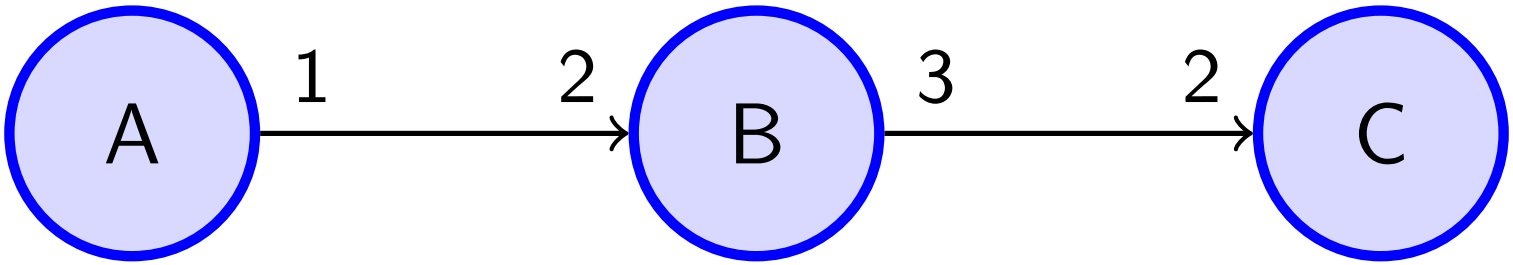
Motivational Example: Streaming Media Application



- If designer is not aware of MoCs, similar application will likely be implemented with
  - RTOS to schedule the processes
  - message queues for process communication
- . . . but this comes with an overhead in form of
  - the RTOS itself
  - safety margin for message queue buffers

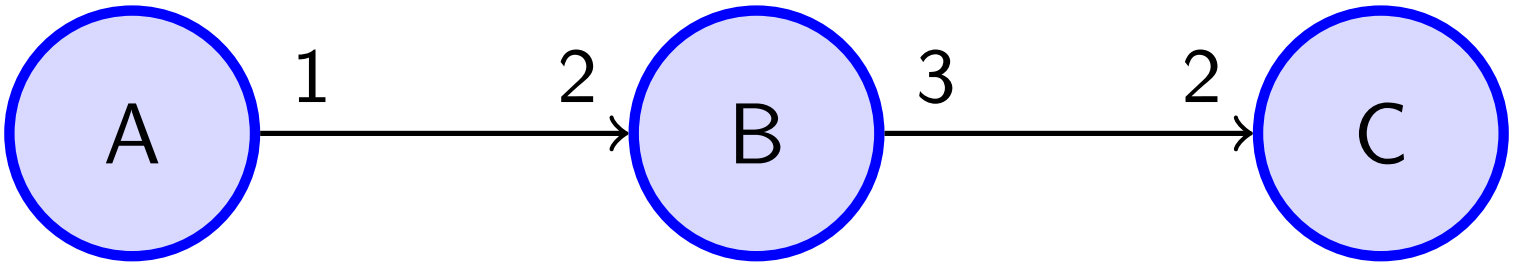
Inefficient implementation!

Motivational Example: Streaming Media Application



- If designer is aware of the synchronous data flow (SDF) MoC [Lee1987]
  - static schedule can be derived: AABCAABCC
  - required buffer size can be implemented: $2 + 4 = 6$

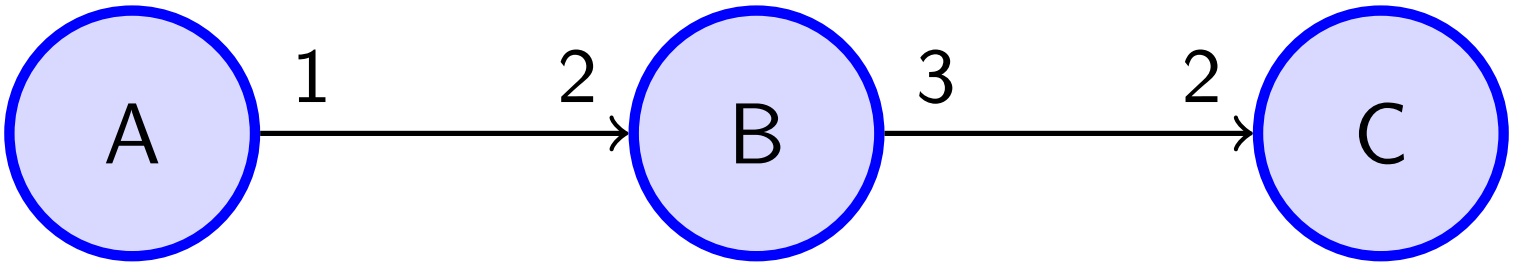
Motivational Example: Streaming Media Application



- If designer is aware of the synchronous data flow (SDF) MoC [Lee1987]
  - static schedule can be derived: AABCAABCC
  - required buffer size can be implemented: $2 + 4 = 6$

- Efficient implementation!

Motivational Example: Streaming Media Application



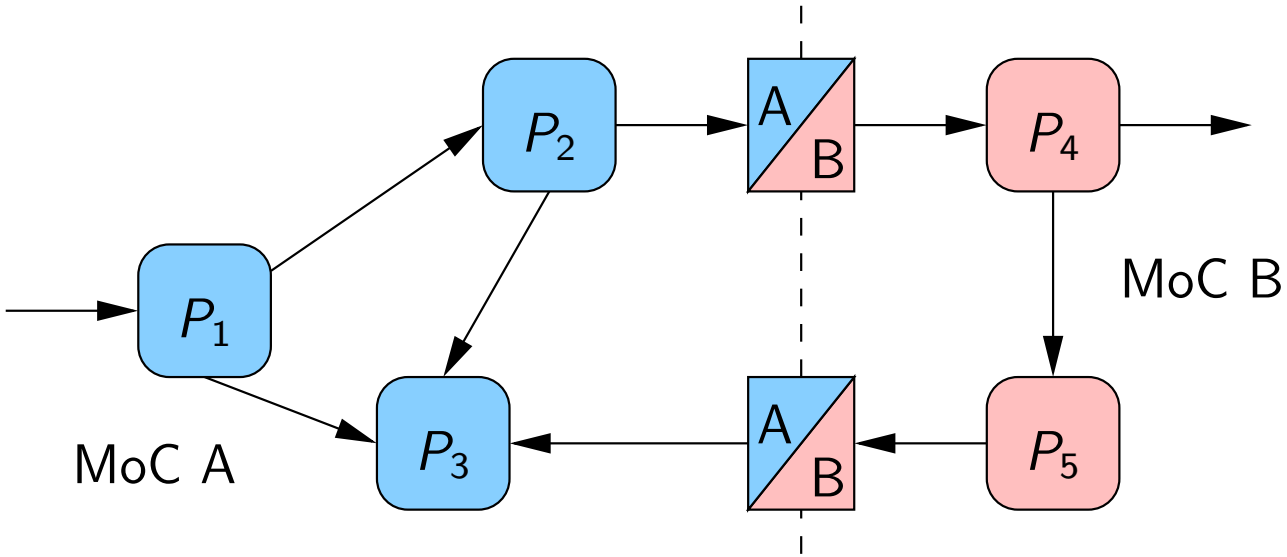
- If designer is aware of the synchronous data flow (SDF) MoC [Lee1987]
  - static schedule can be derived: AABCAABCC
  - required buffer size can be implemented: $2 + 4 = 6$

- Efficient implementation!
- . . . but usual designers are not aware of MoC theory

ForSyDe (Formal System Design)

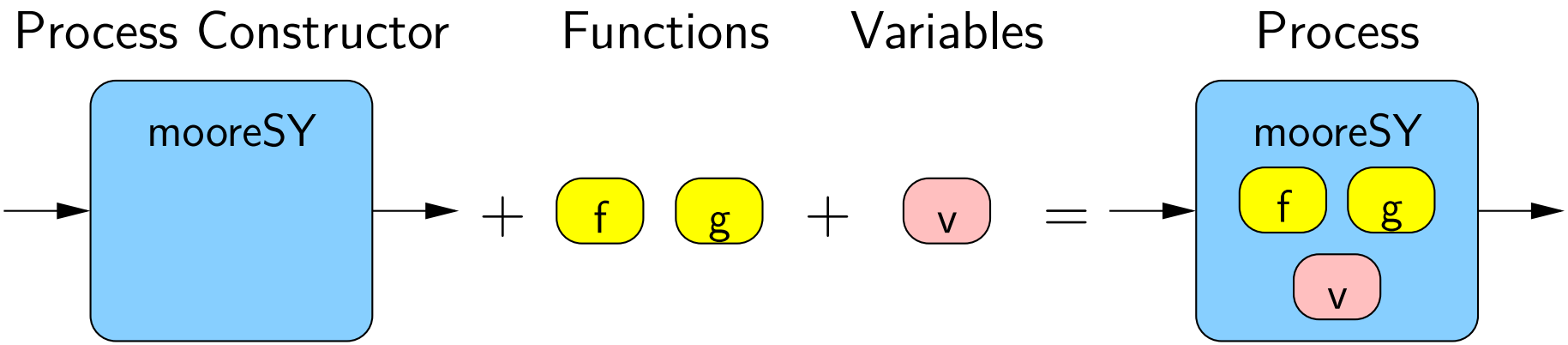
System Model

- A system is modeled as concurrent process network
- Processes belonging to different models of computation communicate via MoC interfaces
- ForSyDe libraries in Haskell and SystemC to support the designer to create a formal model and exist for several MoCs
  - e.g. synchronous MoC, continuous time MoC, SDF MoC



Designing in ForSyDe: Processes

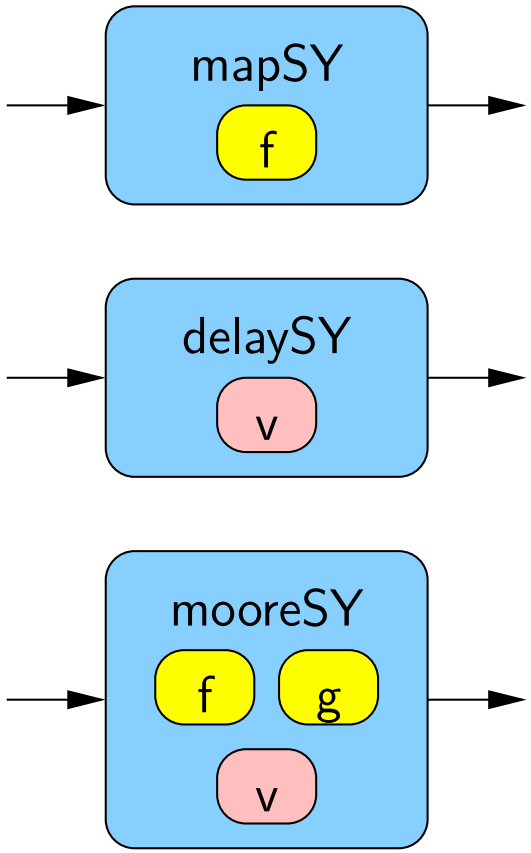
- A process is always designed by means of a process constructor
- Process constructor defines communication interface of process
- Process constructor takes side-effect-free functions and values as arguments and returns deterministic process



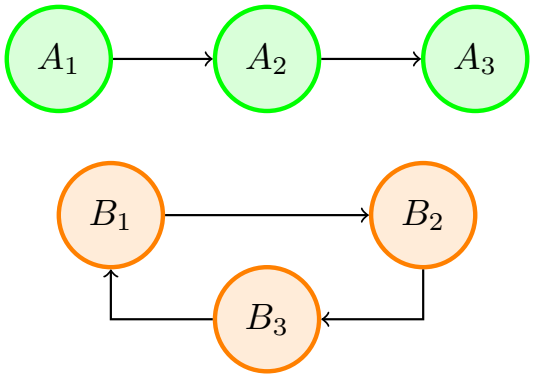
Designing in ForSyDe: Process Constructor Benefits

The concept of process constructor

- separates communication and computation
  - Communication and interface: process constructor
  - Computation: function
- forces designer to develop structured formal model enable formal analysis
- allows to define 'mappings' to implementation languages



ForSyDe Software Design Flow

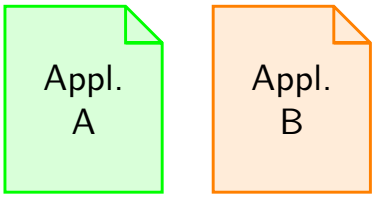


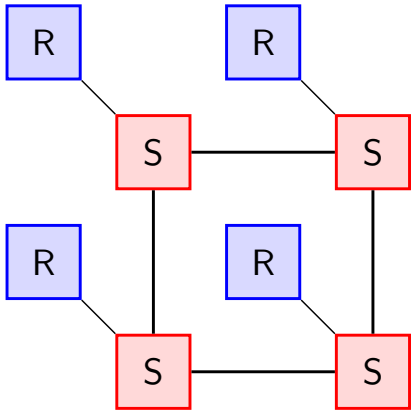
Analyzable Application Models

- formal base (MoCs)
- executable

Design Constraints

- real-time
- power and energy
- ...



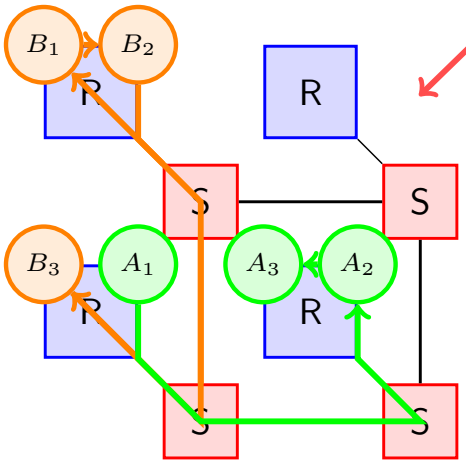


Platform Architecture

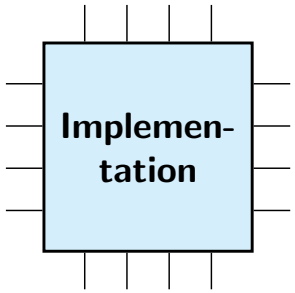
- multiprocessor
- predictable performance

Design Space Exploration
(based on formal model and predictable architecture)

Mapping



Synthesis and Compilation



Implementation

- Customized hardware
- Efficient software

The SAFEPOWER Project

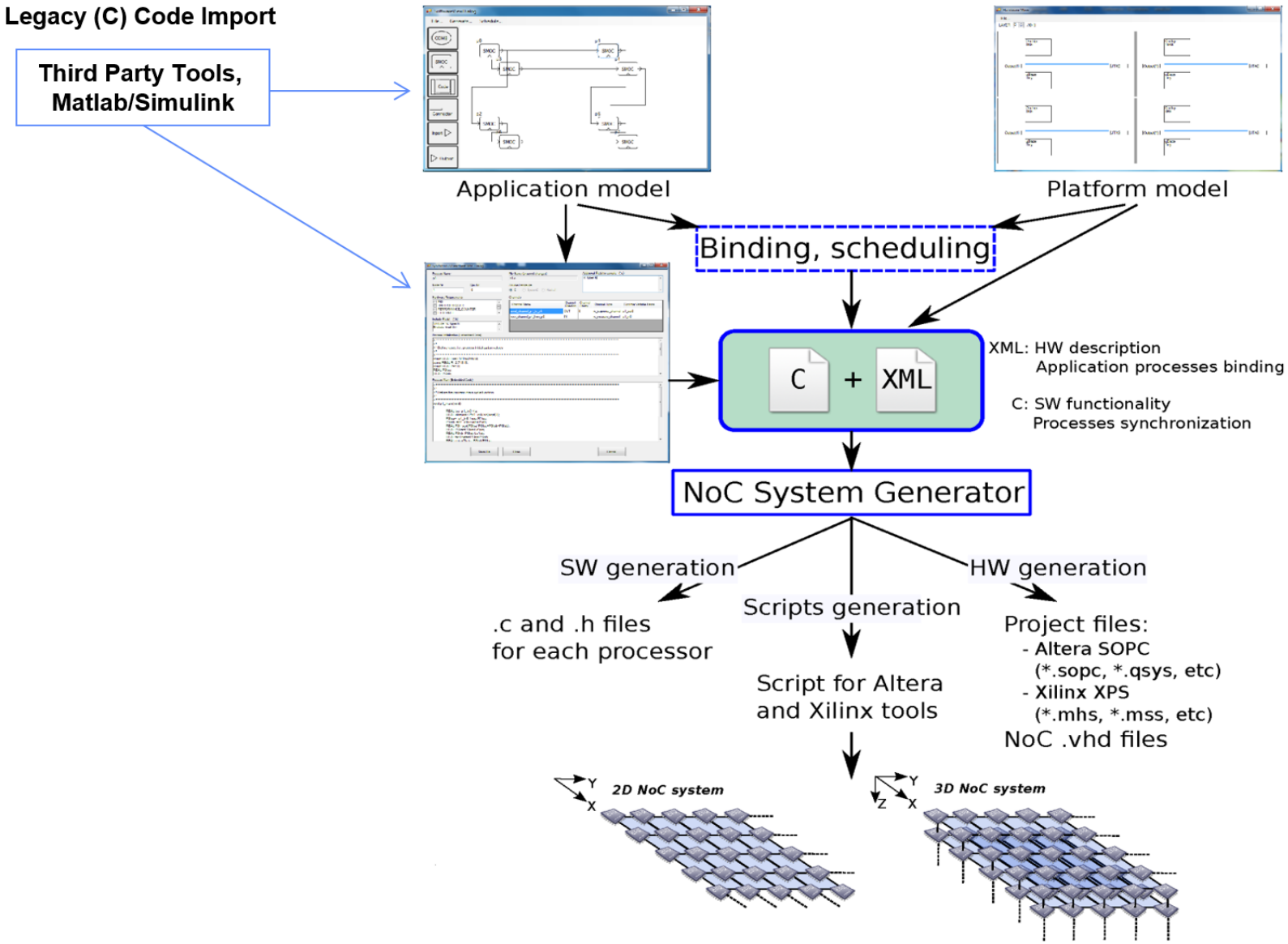
- Funded by EU's Horizon 2020 research and innovation programme
- Objective: Development of low power mixed-criticality systems through the provision of a reference architecture, platforms and tools to facilitate the development, testing, and validation of these kinds of systems
- Eight partners from academia and industry
- Swedish partners: Saab and KTH
- Duration: 2016-2018 (36 months)
- Project web page: `https://www.uni-siegen.de/nt/safepower/`

SAFEPOWER

SAFEPOWER-related research

1. Design of predictable network-on-chip with low-power features
2. Integration into NoC System Generator
3. Extension of design space exploration tool to target 'SAFEPOWER-NoC'
4. Mixed-criticality avionics case study
   - System modelling with ForSyDe and design space exploration
   - Implementation on FPGAs using NoC System Generator
   - Connected FPGA-platforms (IMA-concept)
   - Avionics applications of mixed-criticality

NoC System Generator



Links to Further Information

- ForSyDe web page
  - `https://forsyde.ict.kth.se/`
- Github link to publicly available ForSyDe tools and libraries
  - `https://github.com/forsyde`
- NoC System Generator web page
  - `https://forsyde.ict.kth.se/noc_generator/`
- SAFEPOWER project web page
  - `https://www.uni-siegen.de/nt/safepower/`

Thanks for your attention!