Exploring Power and Throughput for Dataflow Applications on Predictable NoC Multiprocessors

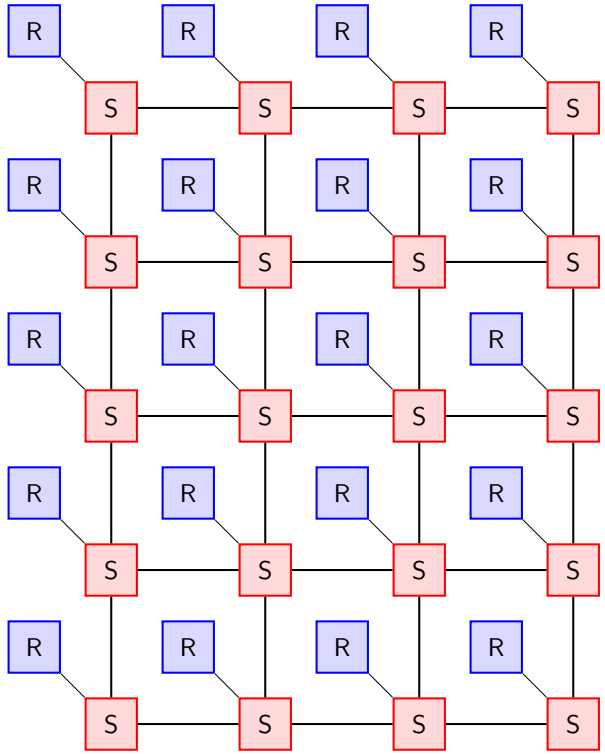
Kathrin Rosvall, **Tage Mohammadat***, George Ungureanu, Johnny Öberg, Ingo Sander
Department of Electronics
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology
Stockholm, Sweden
*tagem@kth.se

Outline

Design Challenge

Timing Guarantees in the Many-Core Era

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

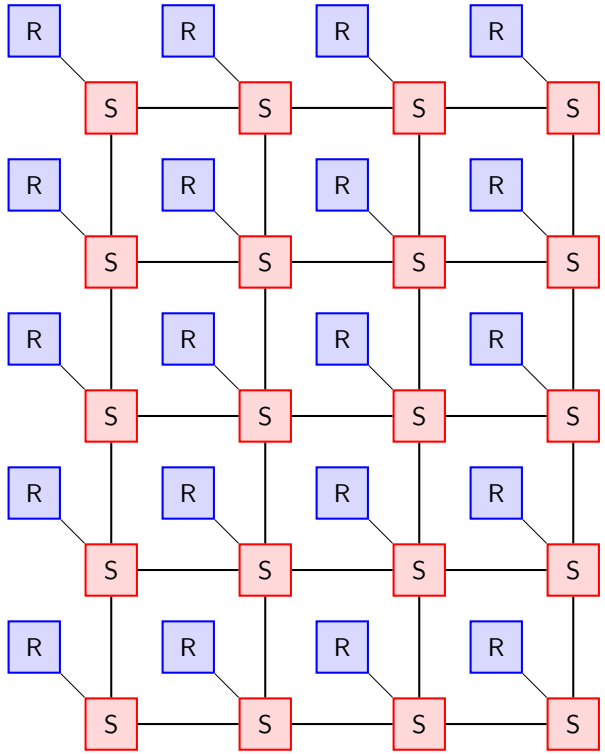


Network-on-Chip (NoC)

Design Challenge

Timing Guarantees in the Many-Core Era

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



Network-on-Chip (NoC)

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

Design Challenge

Mixed-criticality challenge for Transport Industry

Hard requirements on timing.

Typical use-cases require low power consumption and mixed-critical.

- Simple use-cases have hundreds to millions points in design space.
- Dependencies between software and hardware exacerbate complexity.

Design Challenge

Mixed-criticality challenge for Transport Industry

Hard requirements on timing.

Typical use-cases require low power consumption and mixed-critical.

- Simple use-cases have hundreds to millions points in design space.
- Dependencies between software and hardware exacerbate complexity.

Consequence

- Architectural designs are experience-based and not methodological.
- Large safety-margins at the cost of efficiency.
- Engineering costs, design and verification time, are exploding.

Design Challenge

Mixed-criticality challenge for Transport Industry

Hard requirements on timing.

Typical use-cases require low power consumption and mixed-critical.

- Simple use-cases have hundreds to millions points in design space.
- Dependencies between software and hardware exacerbate complexity.

Consequence

- Architectural designs are experience-based and not methodological.
- Large safety-margins at the cost of efficiency.
- Engineering costs, design and verification time, are exploding.

Vision

Automated design methods that are correct-by-construction!

ForSyDe (Formal System Design) Vision

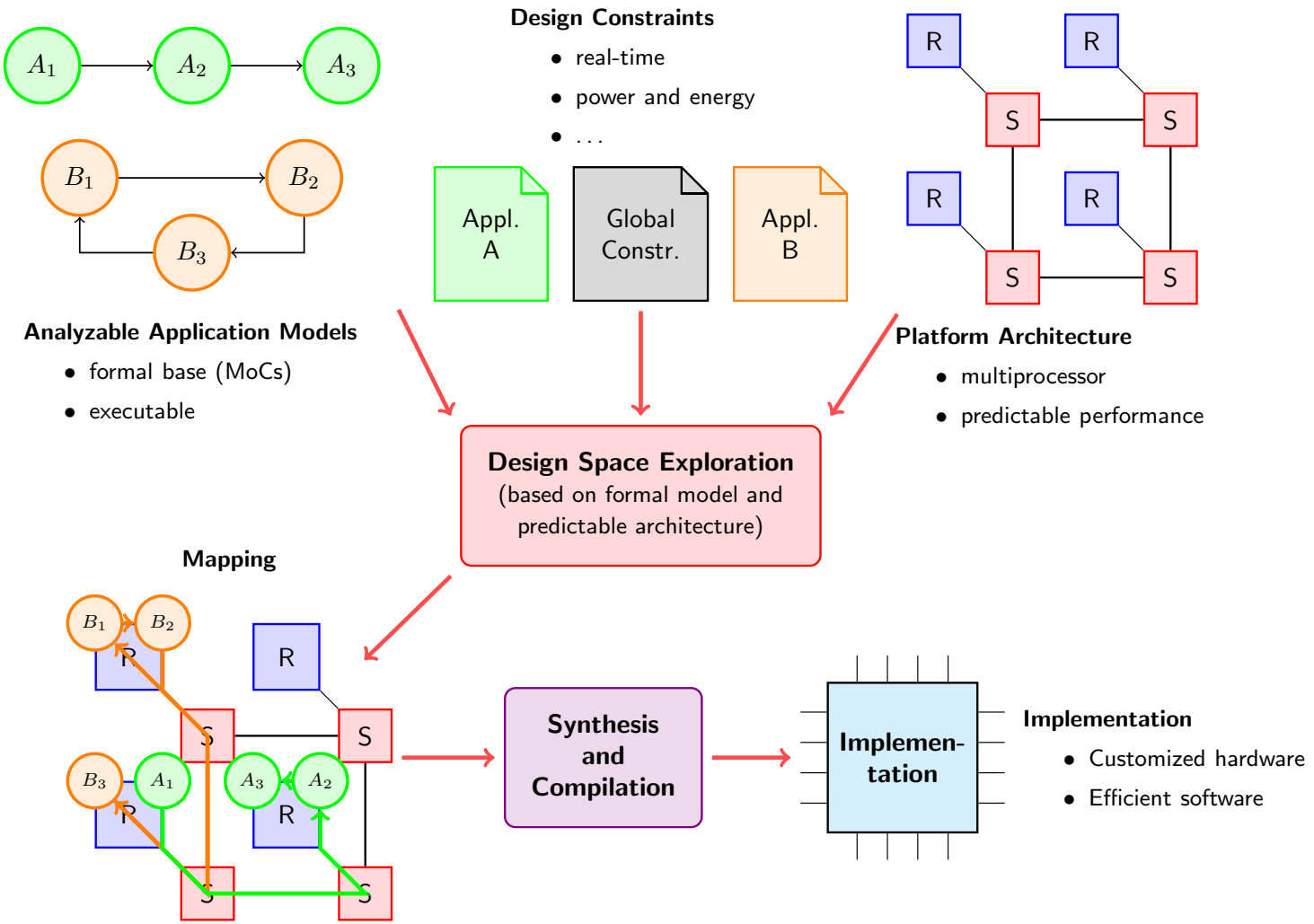
The approach

ForSyDe envisions a correct-by-construction design flow by combining

- System modelling based on models of computation
- Computing platforms providing tight computational metrics, e.g.:
  - computation and communication time
  - communication time
  - ...
- Leveraging
  - Modelling libraries
  - Simulation methods
  - Transformation and refinement rules
  - **Design Space Exploration**
  - Deterministic computing architectures
  - Sound compilation and synthesis methods
  - ...

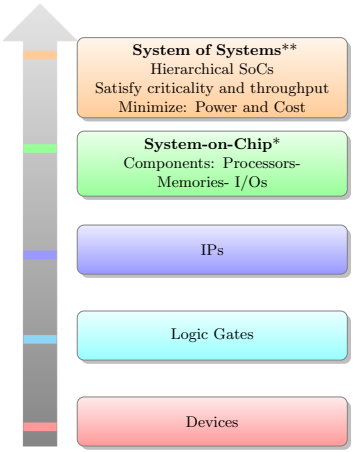
Envisioned Design Flow

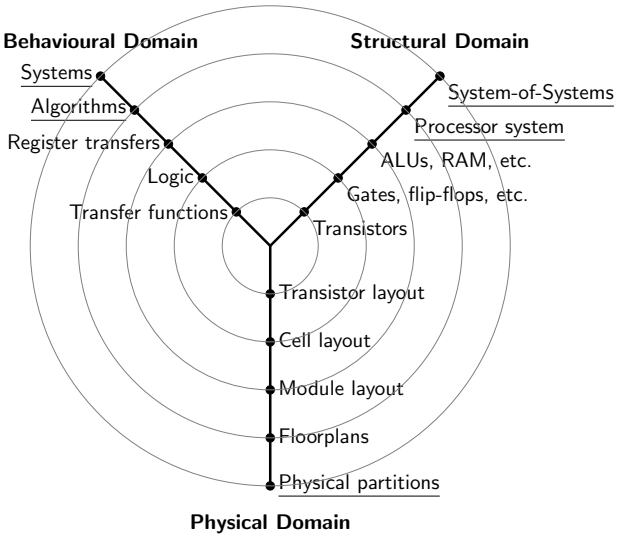
The big picture

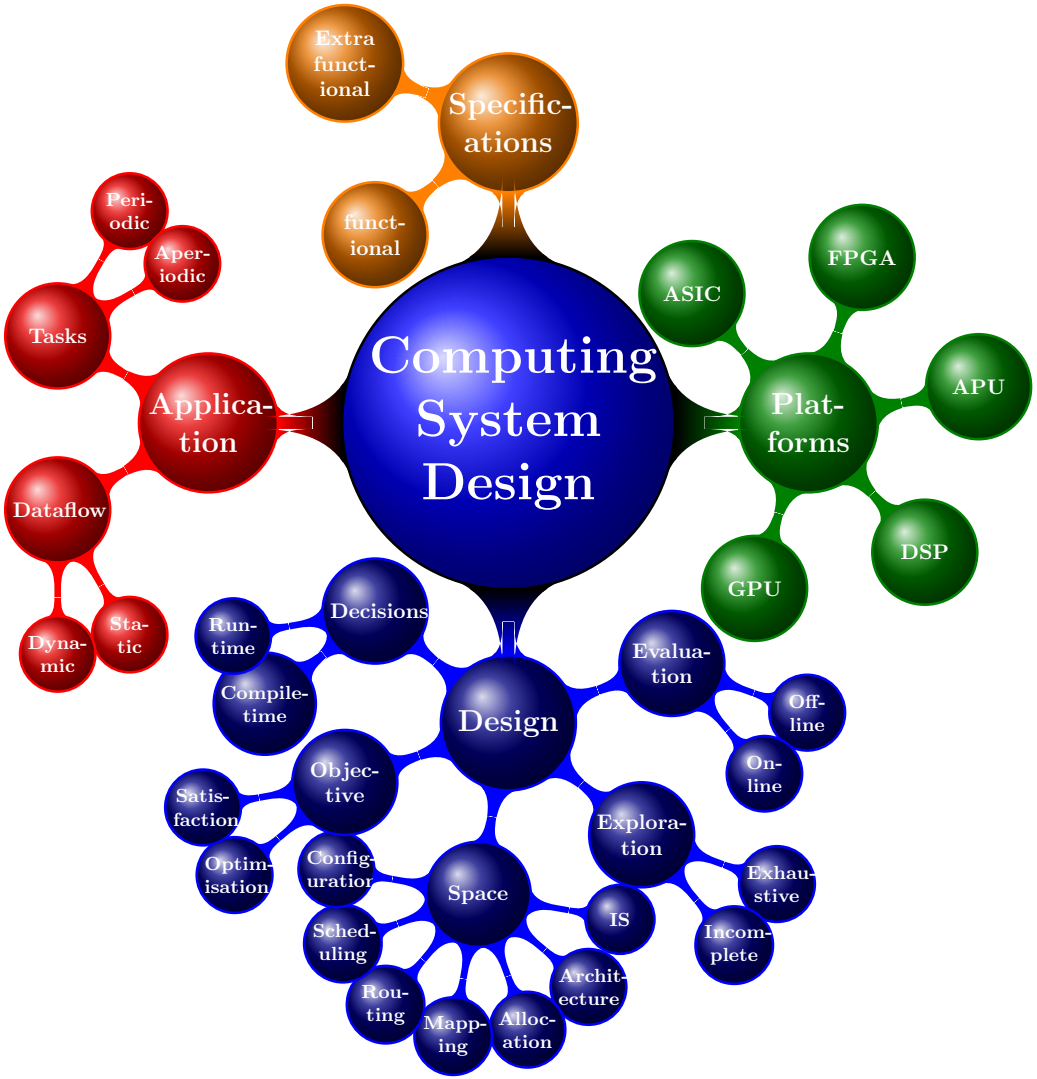


Envisioned Design Flow

Scope and Context





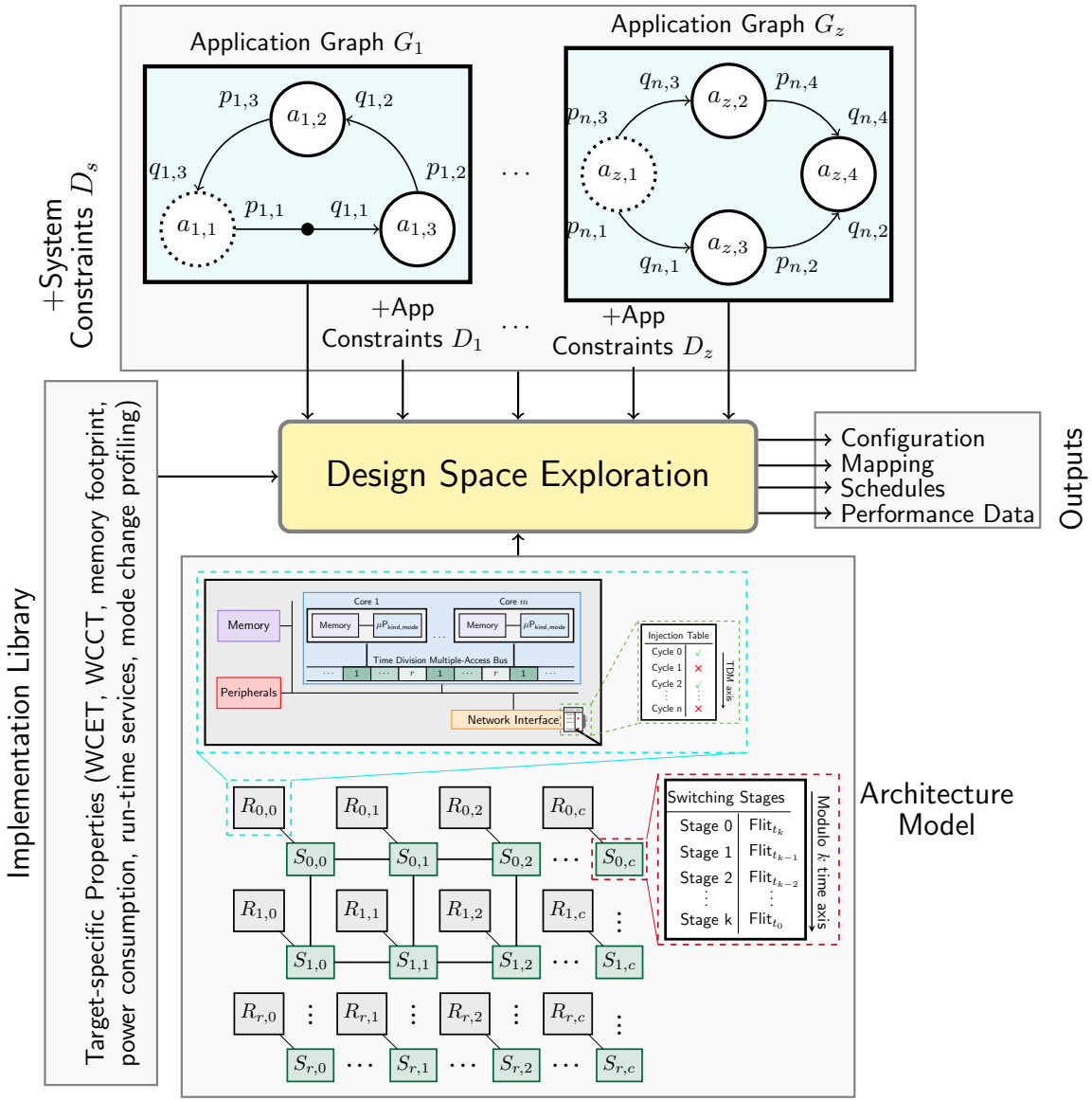


Outline

Contributions

- Support for low-power design of mixed-critical applications on NoC-based MPSoC by:
  - spatio-temporal partitioning of computing and networking resources, while
  - *satisfying* real-time constraints (throughput), and
  - *minimising* system's power consumption.
- Exploiting:
  - applications' formal models
  - platform's heterogeneity, modes and options.
  - temporally-disjoint networks property.

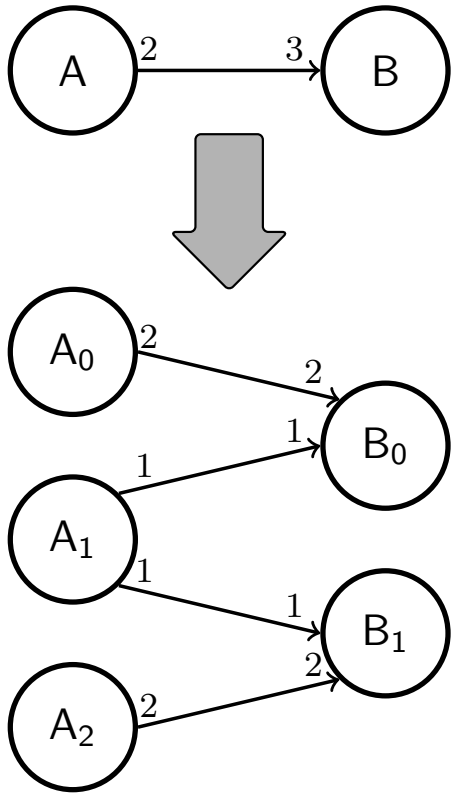
The Framework



The Framework

Application Model

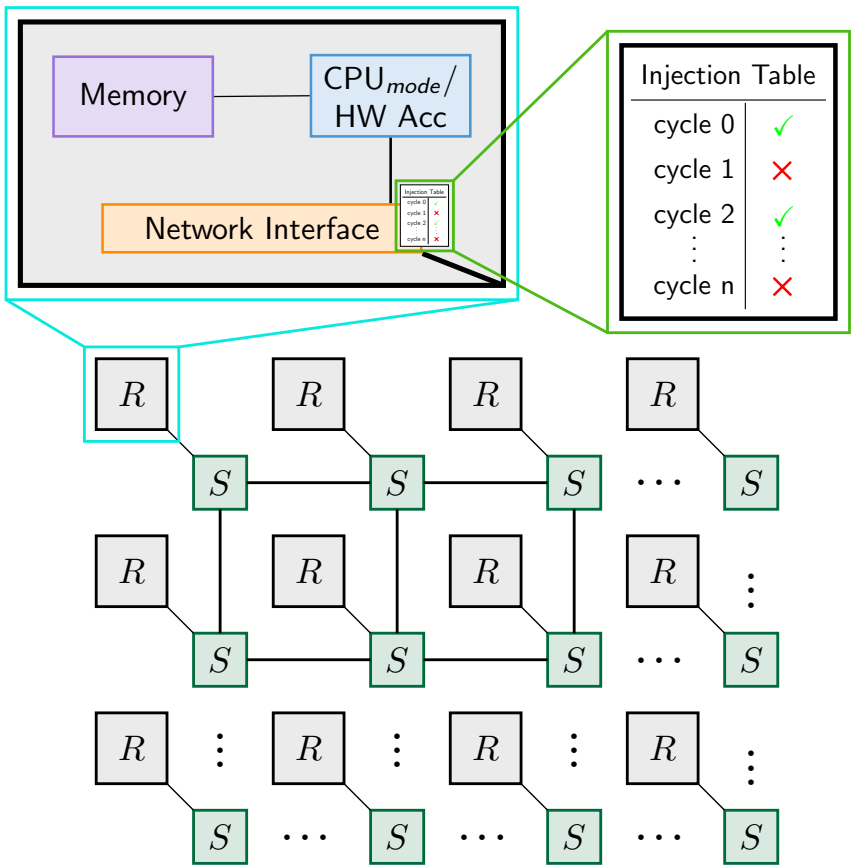
- synchronous dataflow graphs
- actor characterisation library for each processing element type and mode
  - WCET
  - Memory footprint



The Framework

Architectural Model

- Mesh topology, with temporarily-disjoint networks
  - bufferless switching
  - fixed routing (X-,Y-,Z-)
  - guaranteed services via time slot assignment
- Low power and small footprint



Nostrum Network-on-Chip

The Framework

Performance Models

- temporal analysis for applications based on MoC theories.
- linear power models:
  - static: as a function of platform components' kind and mode.
  - dynamic: as a function of resource utilisation/traffic.

$$\begin{aligned}
\texttt{system_power} &= \Big(\sum_{p \in P} \texttt{dynPower}_p + \texttt{statPower}_p\Big) \\
&\quad + \texttt{dynPower}_{NoC} + \texttt{statPower}_{NoC} \\
\texttt{dynPower}_p &= \Big(\sum_{a \in \mathcal{A} | \texttt{proc}_a = p} \texttt{wcet}_a(p, \texttt{proc_mode}_p)\Big) \\
&\quad \cdot \texttt{dynPow}(\texttt{proc_mode}_p) \div \texttt{period}_a \\
\texttt{dynPower}_{NoC} &= \texttt{dynPower}_{NIs} + \texttt{dynPower}_{switches} \\
&\quad + \texttt{dynPower}_{links}
\end{aligned}$$

The Framework

Design Constraints

- Support for mixed-critical applications through:
  - spatio-temporal partitioning of computing and networking resources.
  - time constraints (throughput): *satisfy*
  - low power execution: *minimize*

The Framework

Design Constraints

- Support for mixed-critical applications through:
  - spatio-temporal partitioning of computing and networking resources.
  - time constraints (throughput): *satisfy*
  - low power execution: *minimize*
- Currently supported performance metrics:
  - throughput / iteration period
  - energy consumption
  - others: area and monetary cost, memory consumption

The Method

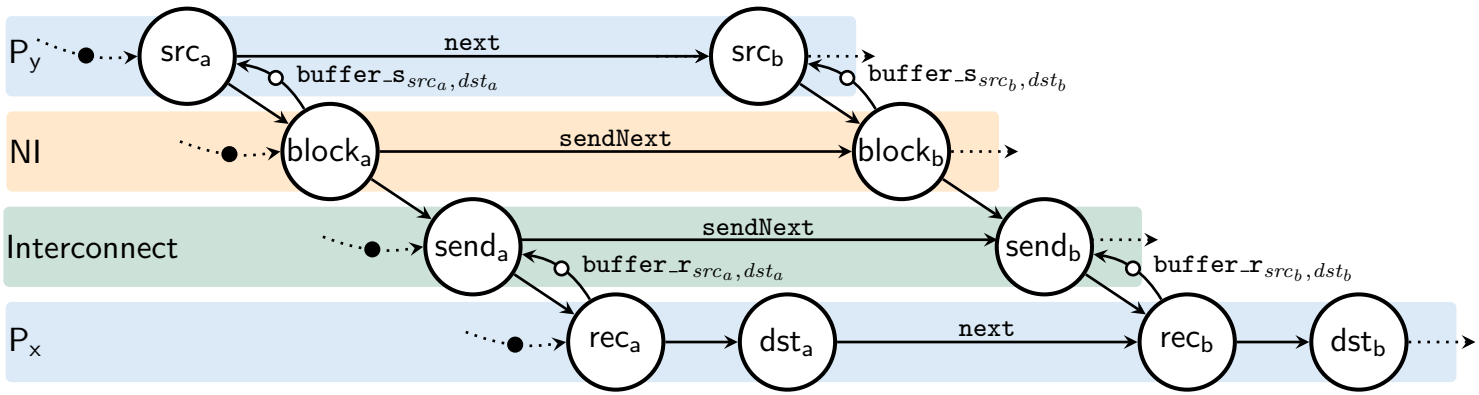
Constraint Programming (CP) Programme

- Captures the problem as a whole
  - no decomposition into sub-problems:
    (potentially) **optimal**, **complete** and **trade-off-aware**
  - Simultaneous mapping, scheduling, configuration and performance analysis
- Declarative nature: **flexible**, **modular** and **extendable model**
  - Supports different design goals with the same CP model
- Separation of concerns: modelling vs solving
  - Complete or heuristic search using the same CP model

The Method

CP model + Dataflow Analysis

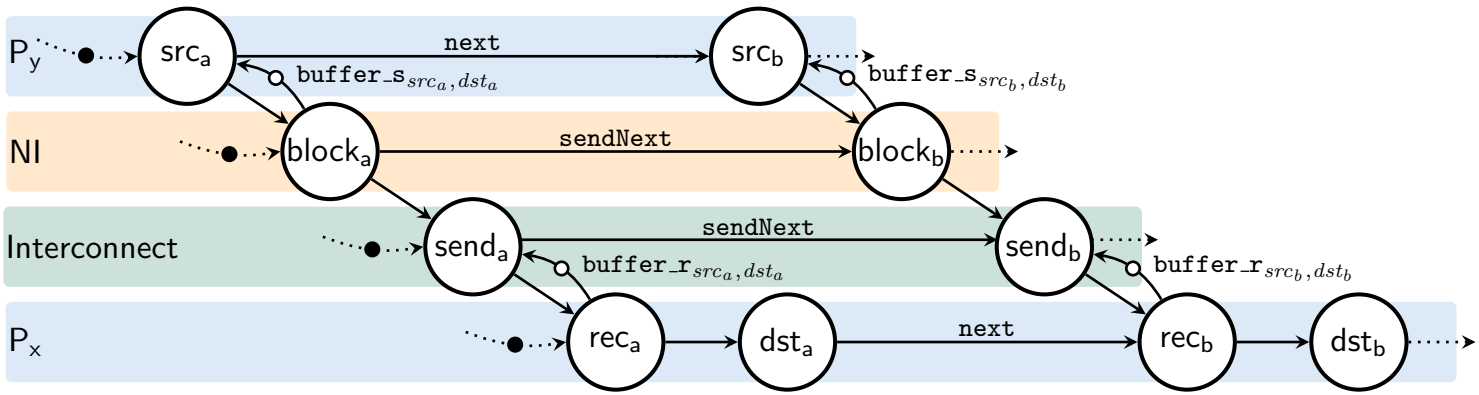
- A CP model reflects a mapping- and scheduling-aware graph (MSAG):
  - captures applications and platform.



The Method

CP model + Dataflow Analysis

- A CP model reflects a mapping- and scheduling-aware graph (MSAG):
  - captures applications and platform.
  - analyses implications of mapping and scheduling (e.g. power).



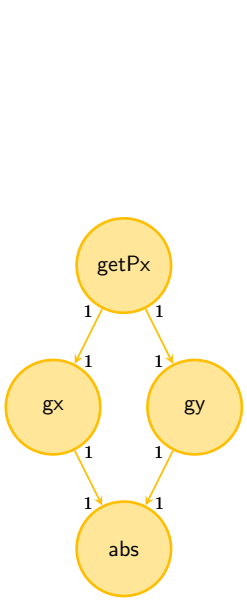
Experiments

Experiments set

#	Description
1	**Physical experiment: TDN-NoC FPGA platform**
2-5	DSE for **larger problem** with the fixed mapping with different optimization goals:
2	power consumption, one TDN slot/processor
3	power consumption, multiple TDN slots/processor
4	throughput for cyclic graph, one TDN slot/processor
5	throughput for cyclic graph, multiple TDN slot/processor

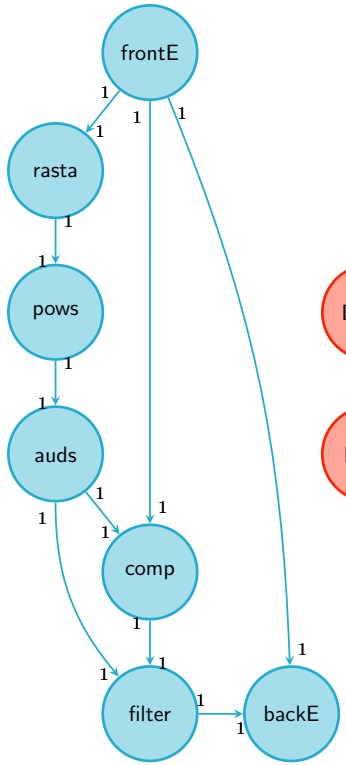
Experiments

Applications set

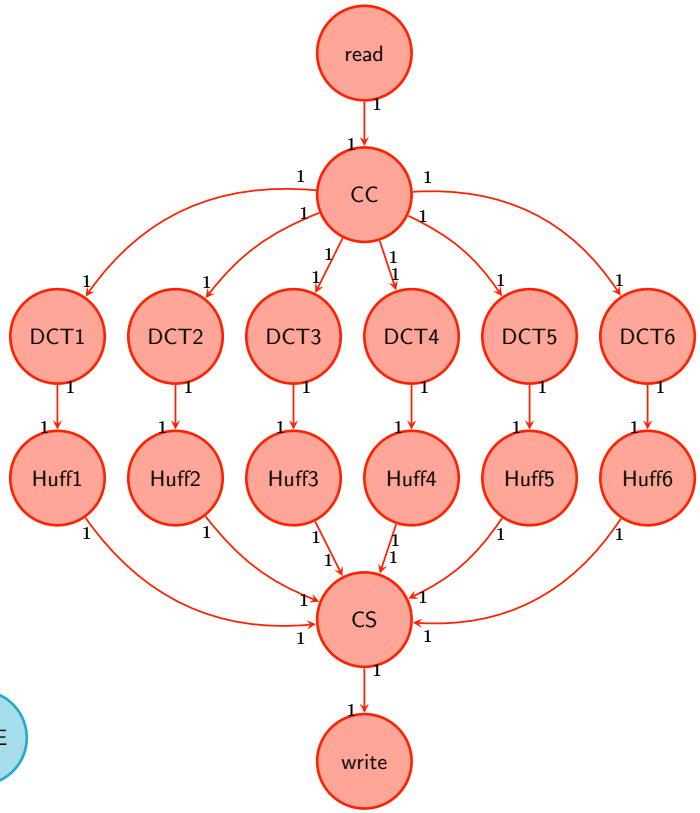


Sobel

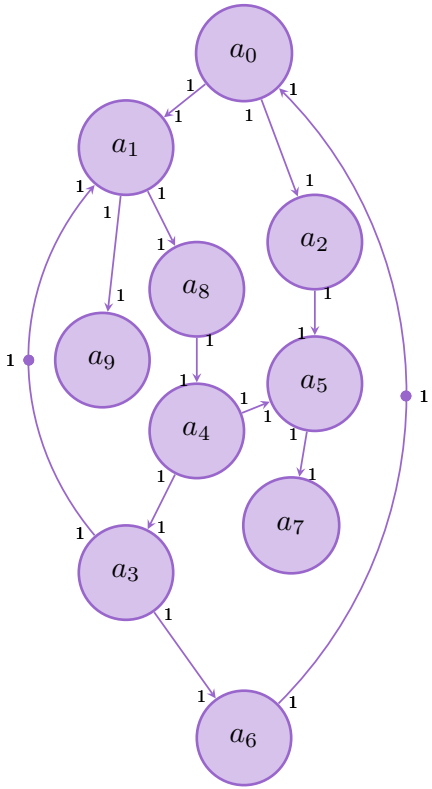
SUSAN



RASTA-PLP



JPEG encoder

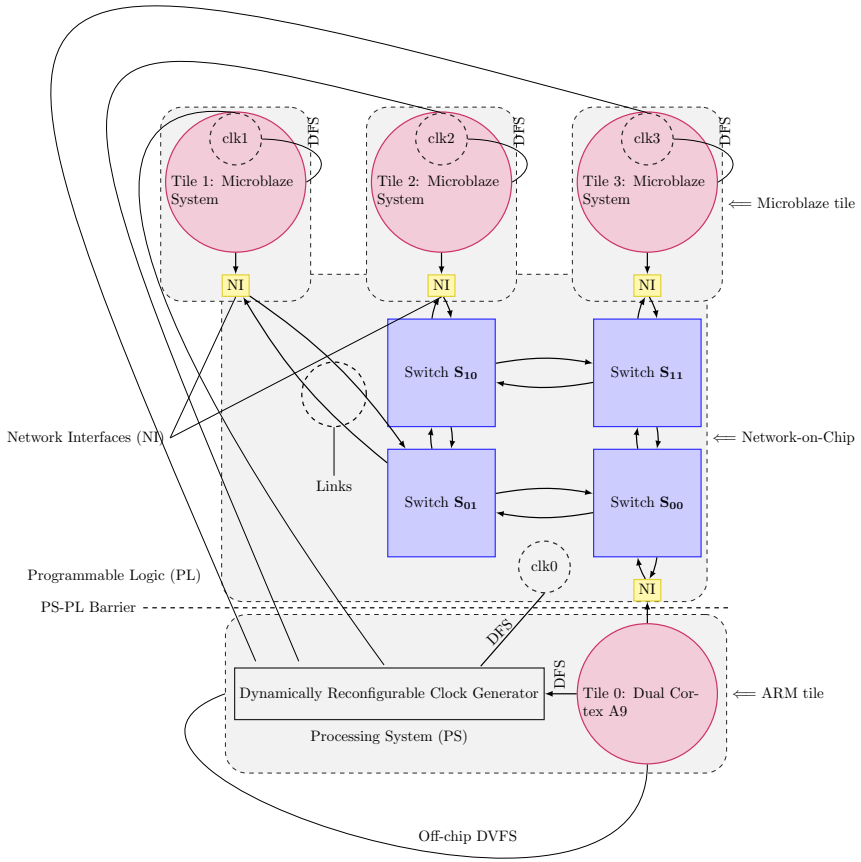


Synthetic Cyclic Graph

Experiment 1: Physical experiment

Validation on a Xilinx Zynq Quadprocessor: Description

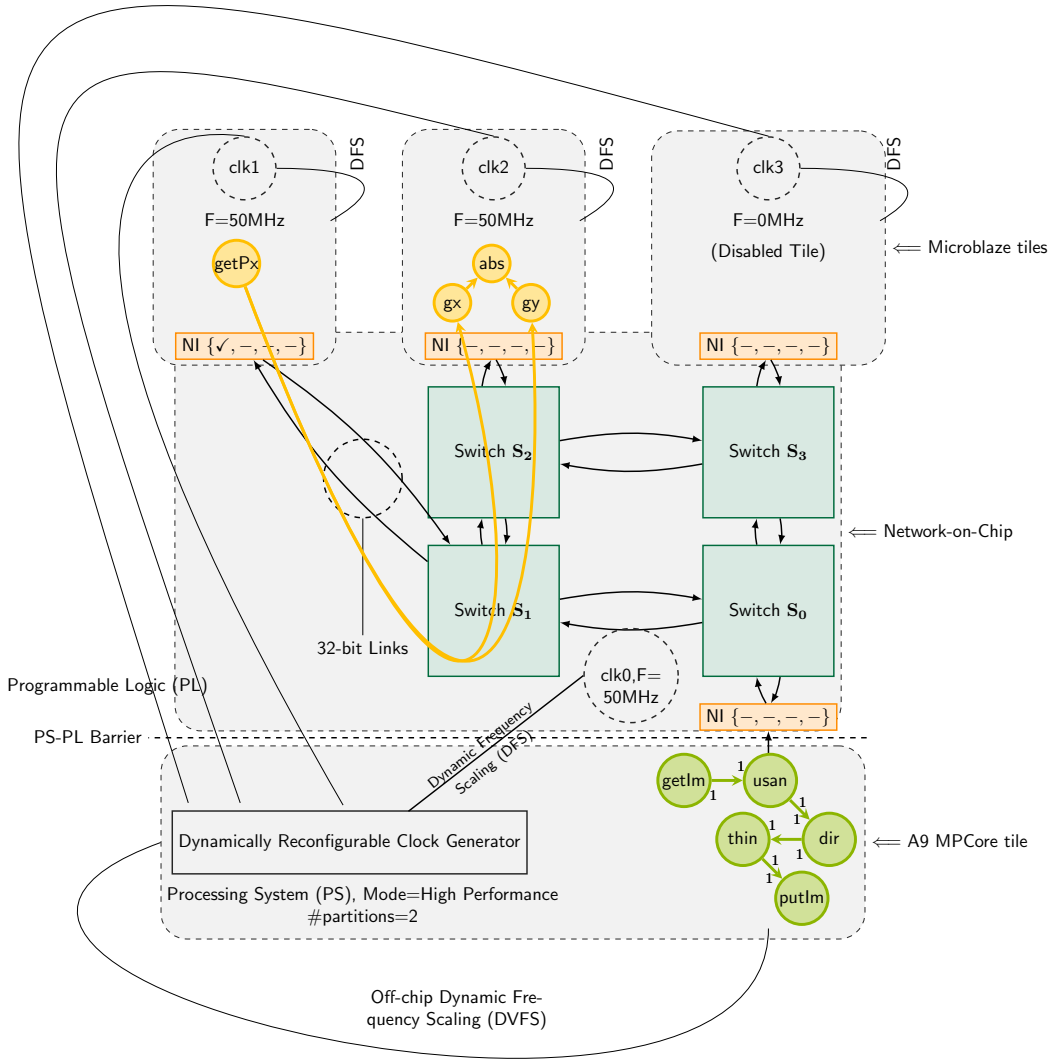
1. Goals: Minimize power consumption + satisfy throughput.
2. App/platform: Sobel & Susan on Nostrum NoC.



Experiment 1: Physical experiment

Validation on a Xilinx Zynq Quadprocessor: Solution

- Allocation
- Configuration
- Mapping
- Scheduling
- Exact area cost
- Caches disabled
- 10% time margin
- Low-Power

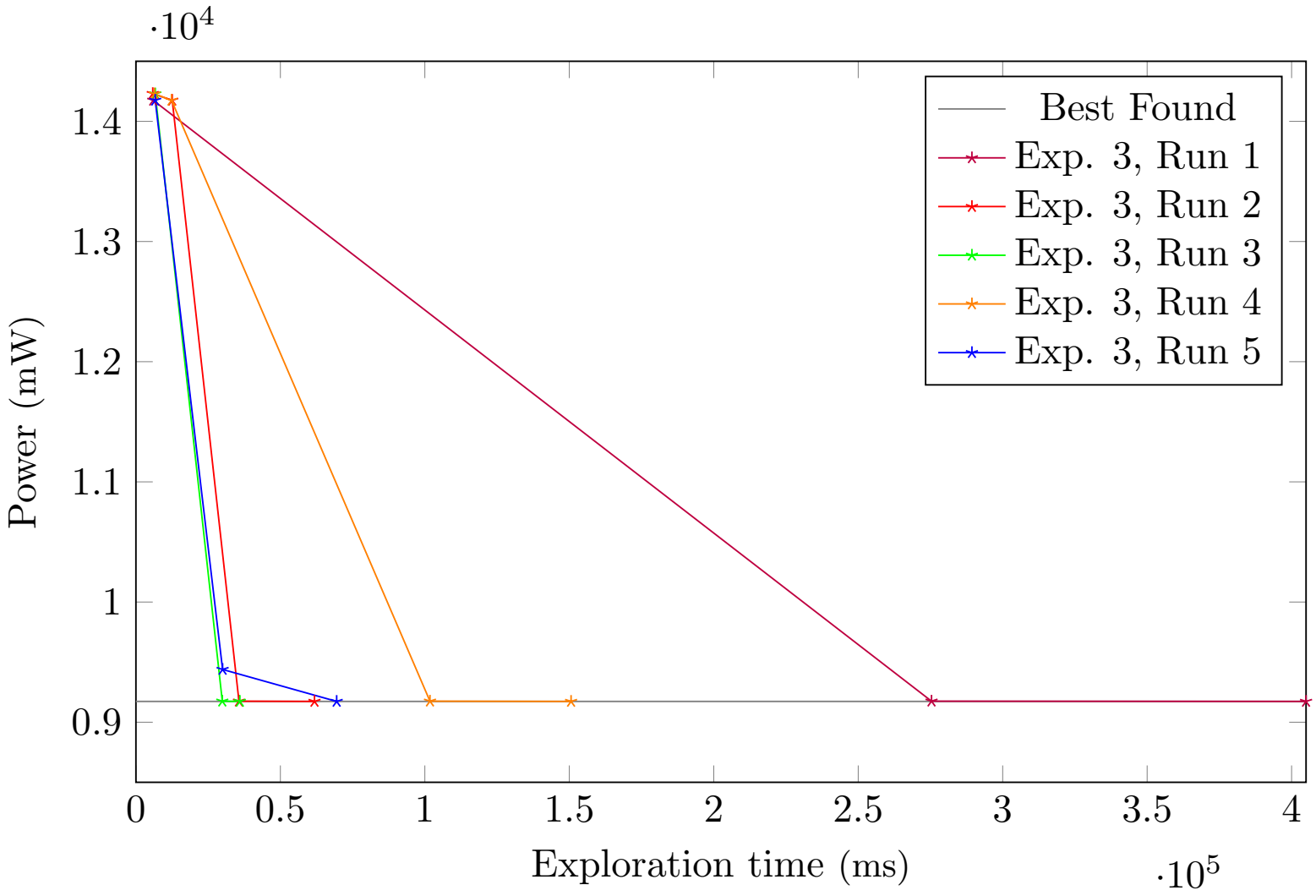


Experiment 3: Larger design

Power Minimisation and Slot Assignment for Fixed Mapping

Experiment 3: Larger design

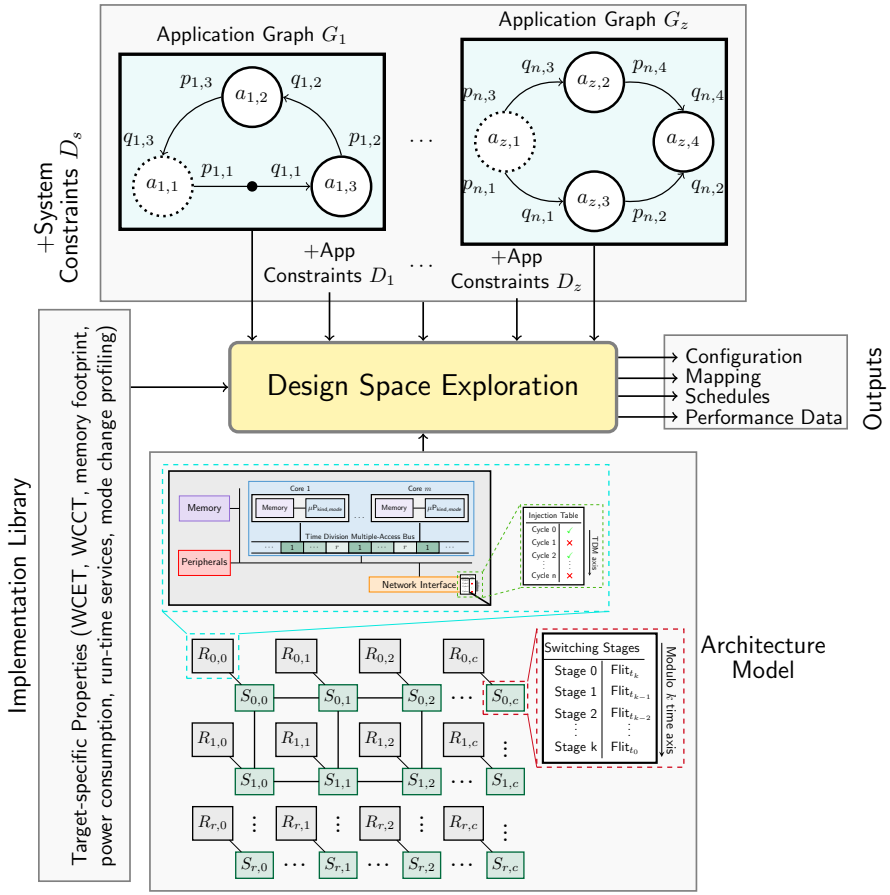
Power Minimisation and Slot Assignment for Fixed Mapping: Solution



Outline

Summary

- Low-power design space exploration of
  - multiple dataflow applications with
  - static mixed-criticality support
  - on a shared MPSoC platform
- support for predictable network-on-chip



Open Problems

- Exploring models-of-computation that change in time.
  - Scenario-aware data-flow graphs.
  - Reconfigurable computing.
- Exploring computer architectures
  - parallelisms within computing elements
  - network topologies
  - memory hierarchy
- Targetting resiliency
  - Graceful degradation
  - Security trade-offs
- Designing for adaptive and distributed computing.
- Distributed exploration: limits 10x10 (Vivado), 6x8 (GeCode).

This work is part of ForSyDe/DeSyDe


Kathrin Rosvall, Tage Mohammadat, George Ungureanu, Johnny Oberg, and Ingo Sander.
Exploring power and throughput for dataflow applications on predictable noc multiprocessors.
In *Euromicro Conference on Digital System Design (DSD 2018)*, Prague, Czech Republic, August 2018.


Kathrin Rosvall and Ingo Sander.
Flexible and trade-off-aware constraint-based design space exploration for streaming applications on heterogeneous platforms.
ACM Transactions on Design Automation of Electronic Systems (TODAES), 23(2), 2017.


Ingo Sander, Axel Jantsch, and Seyed-Hosein Attarzadeh-Niaki.
ForSyDe: System design using a functional language and models of computation.
In *Handbook of Hardware/Software Codesign*. Springer, Dordrecht, 2017.


George Ungureanu and Ingo Sander.
A layered formal framework for modeling of cyber-physical systems.
In *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1715–1720. IEEE, 2017.

- More resources: `https://github.com/forsyde/desyde`