CHAIR FOR COMPILER CONSTRUCTION

cfaed CENTER FOR ADVANCING ELECTRONICS DRESDEN

Programming Heterogeneous Embedded Systems for IoT

Jeronimo Castrillon
Chair for Compiler Construction
TU Dresden
jeronimo.castrillon@tu-dresden.de

Get-together toward a sustainable collaboration in IoT
April 18th, Tunis, Tunisia

Internet of things

Distributed

Real time

Highly heterogeneous

Energy efficient

Resource constrained

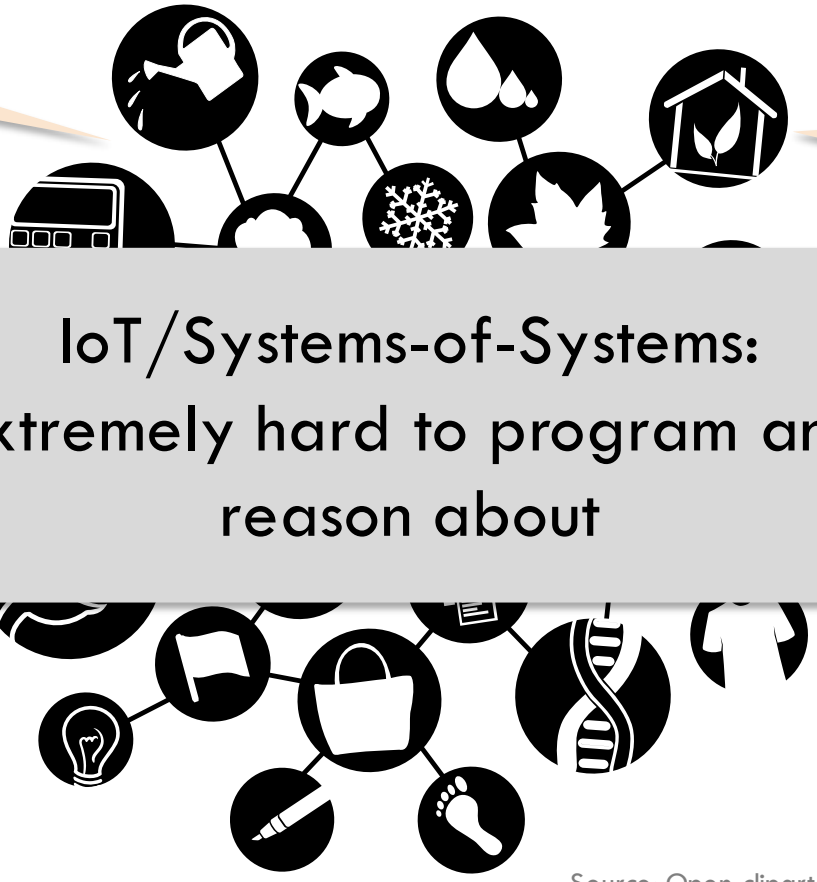
Multiple domains

Source: Open clipart

Internet of things

Distributed

Energy efficient



IoT/Systems-of-Systems: Extremely hard to program and reason about

Real time

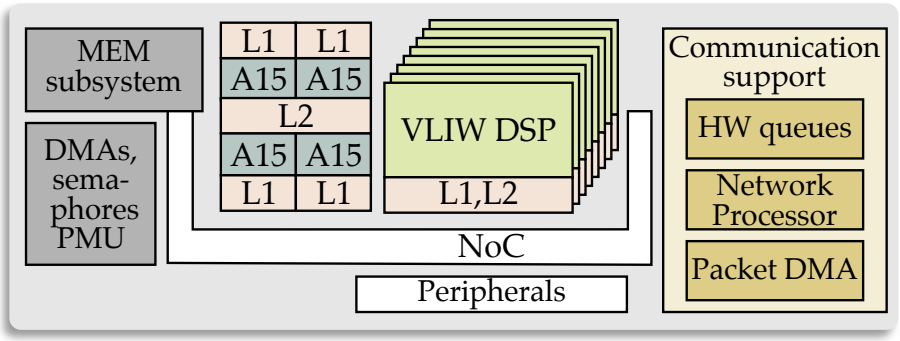
Resource constrained

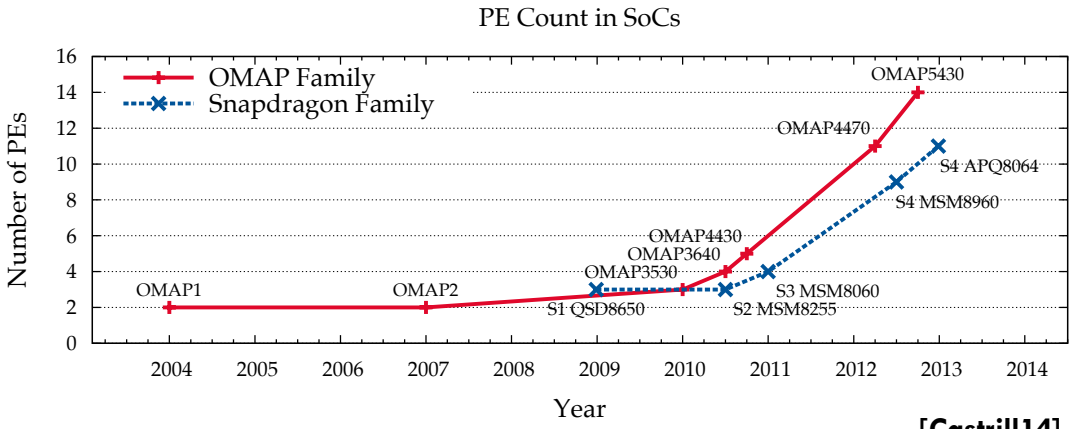
Highly heterogeneous

Multiple domains

Source: Open clipart

Independent nodes: Trends

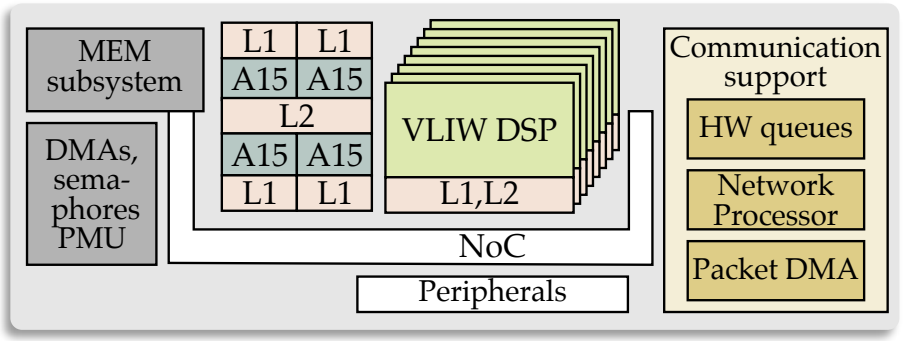


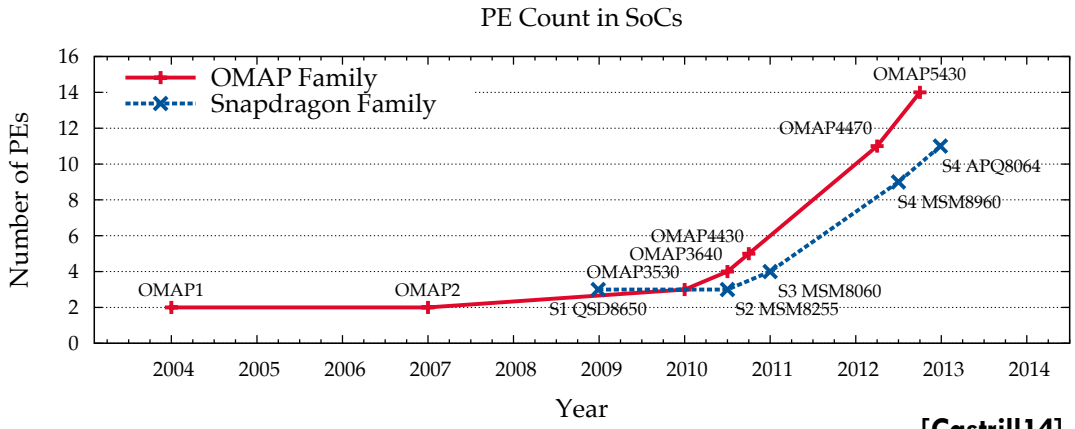


[Castrill14]

- **Single node: HW complexity**
  - Increasing heterogeneity
  - Complex interconnect
  - Increasing number of cores
- **SW "complexity"**
  - Not anymore a simple control loop
  - Need expressive programming models

Independent nodes: Trends





[Castrill14]

- **Single node: HW complexity**
  - Increasing heterogeneit
  - Complex interconnect
  - Increasing number of c
- **SW "complexity"**
  - Not anymore a simple control loop
  - Need expressive programming models

Outline

- Introduction
- Models of computation
- Tool flows
- Outlook for IoT
- Summary

Outline

- Introduction
- **Models of computation**
- Tool flows
- Outlook for IoT
- Summary

What is a MoC?

A **model of computation** is the definition of the set of allowable operations used in computation and their respective costs. It is used for measuring the complexity of an algorithm in execution time and or memory space

Wikipedia: From computational theory

Model of computations are abstract specifications of how a computation can progress

http://c2.com/cgi/wiki?ModelsOfComputation

A **Model of computation** is a collection of rules that govern the interaction of components

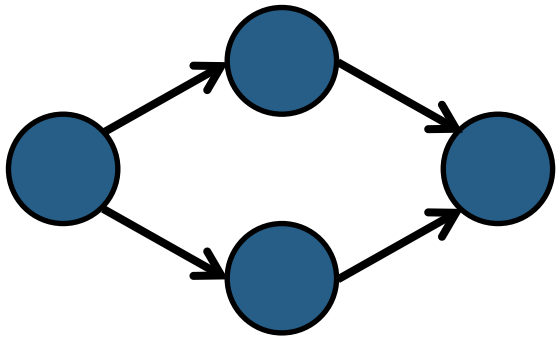
Ptolemaeus, C. (Ed.). System Design, Modeling, and Simulation using Ptolemy II Ptolemy.org, 2014

(Parallel) Models of Computation (MoCs): Rules

A **Model of computation** is a collection of rules that govern the interaction of components

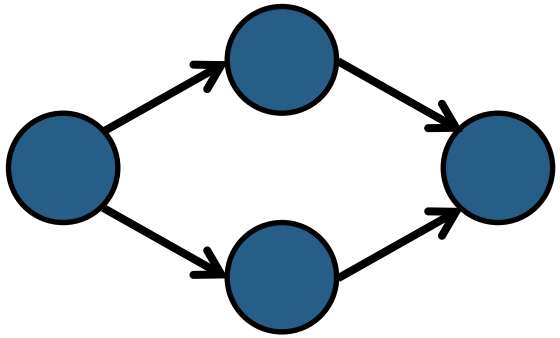
Ptolemaeus, C. (Ed.). System Design, Modeling, and Simulation using Ptolemy II Ptolemy.org, 2014

- What are the components
  - Threads, processes, actors, tasks or procedures
- Execution and concurrency
  - Order and determinism
- Communication mechanisms: How do components exchange data
  - Asynchronous or rendezvous, perfect or lossy

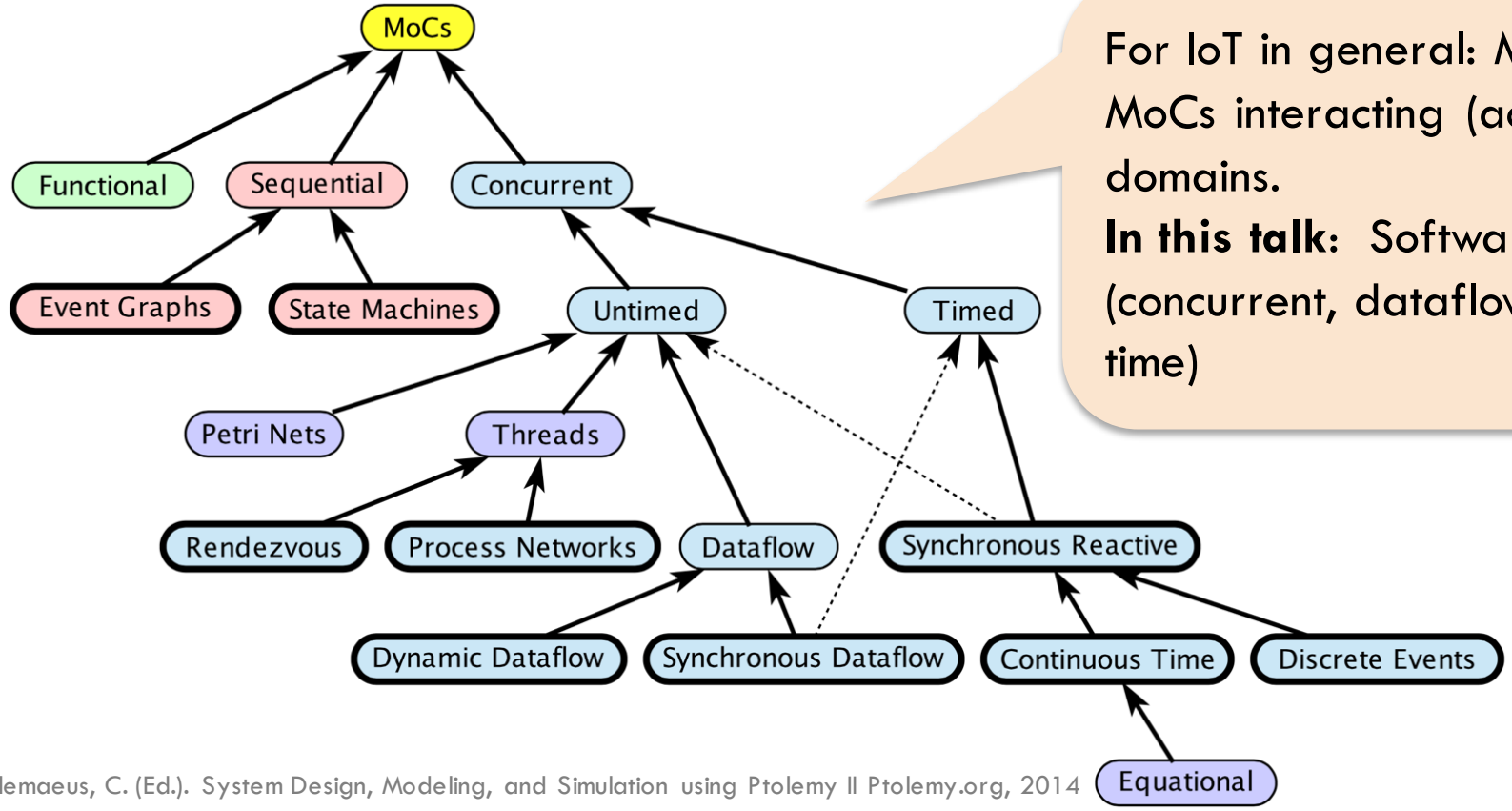


Why MoCs?

- Depending on the rules, MoCs feature different **properties**
- **Properties**
  - Relate to the power: what can be computed with the model
  - Makes it easier for automatic tools to understand and synthesize code
    - E.g., compile-time optimization, support resource constraints, …
- Examples
  - Can the application run on bounded memory?
  - Can we compute the maximal throughput?
  - Is the execution deterministic?



MoCs: Overview



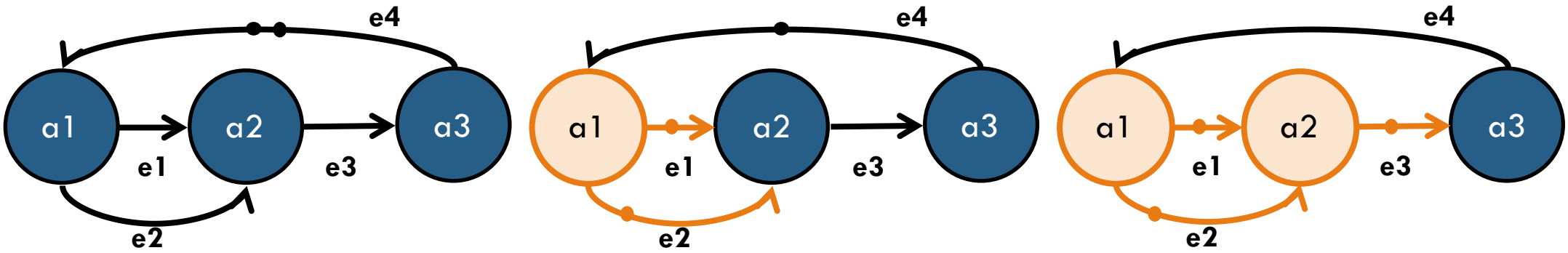
For IoT in general: Mixture of MoCs interacting (across different domains.

In this talk: Software (concurrent, dataflow + PN + time)

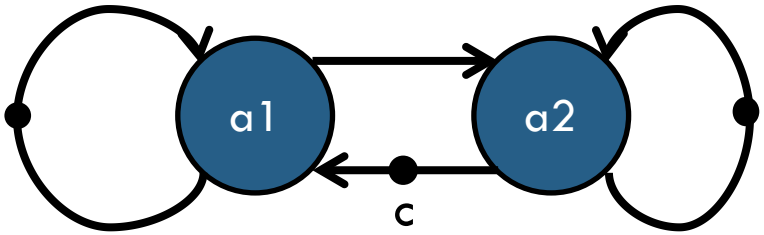
Ptolemaeus, C. (Ed.). System Design, Modeling, and Simulation using Ptolemy II Ptolemy.org, 2014

Homogeneous Synchronous Dataflow (HSDF)

- Graph with simple execution semantics

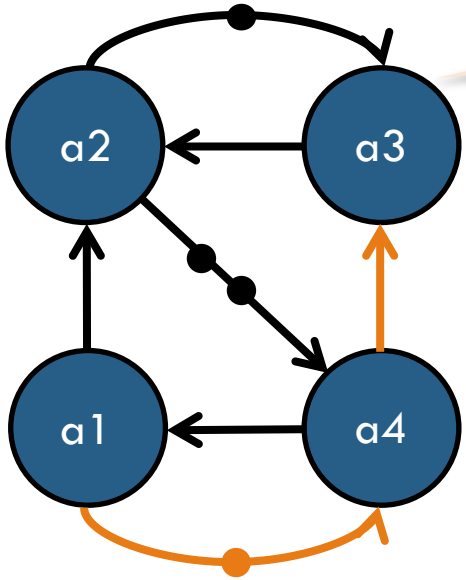


- Functional, graph model can be extended to model
  - Bounded buffers
  - Schedules

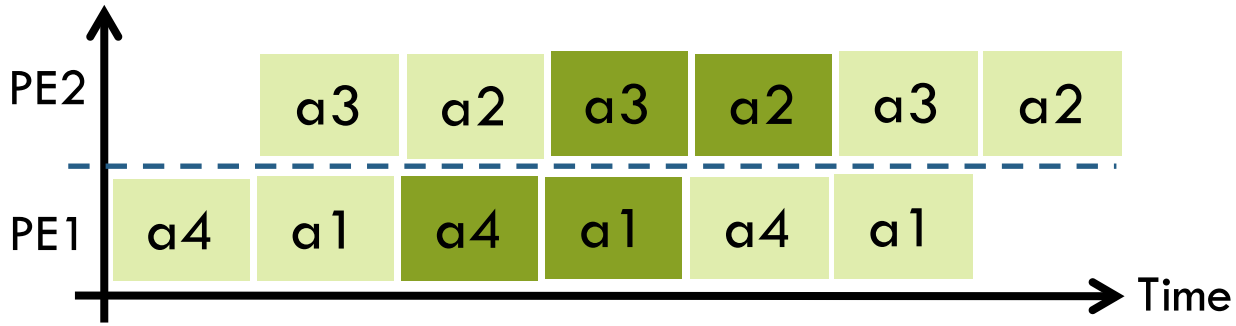


HSDF: schedules

- Overlapping schedule



How to delay the execution of a3?,
how to prevent a4 from executing earlier?

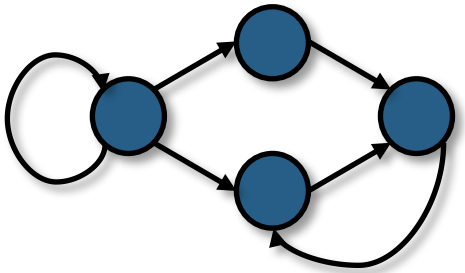


Outline

- Introduction
- Models of computation
- **Tool flows**
- Outlook for IoT
- Summary

Programming flow: Overview

MoC-based Application



Non-functional specification

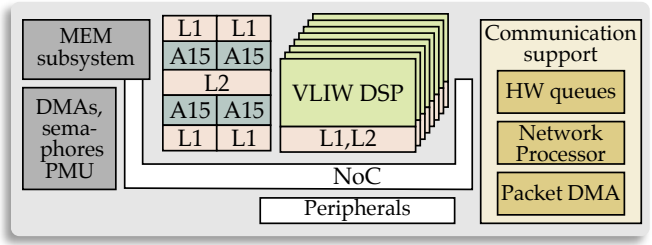
Analysis

Synthesis

Code generation

[Castrill11a]

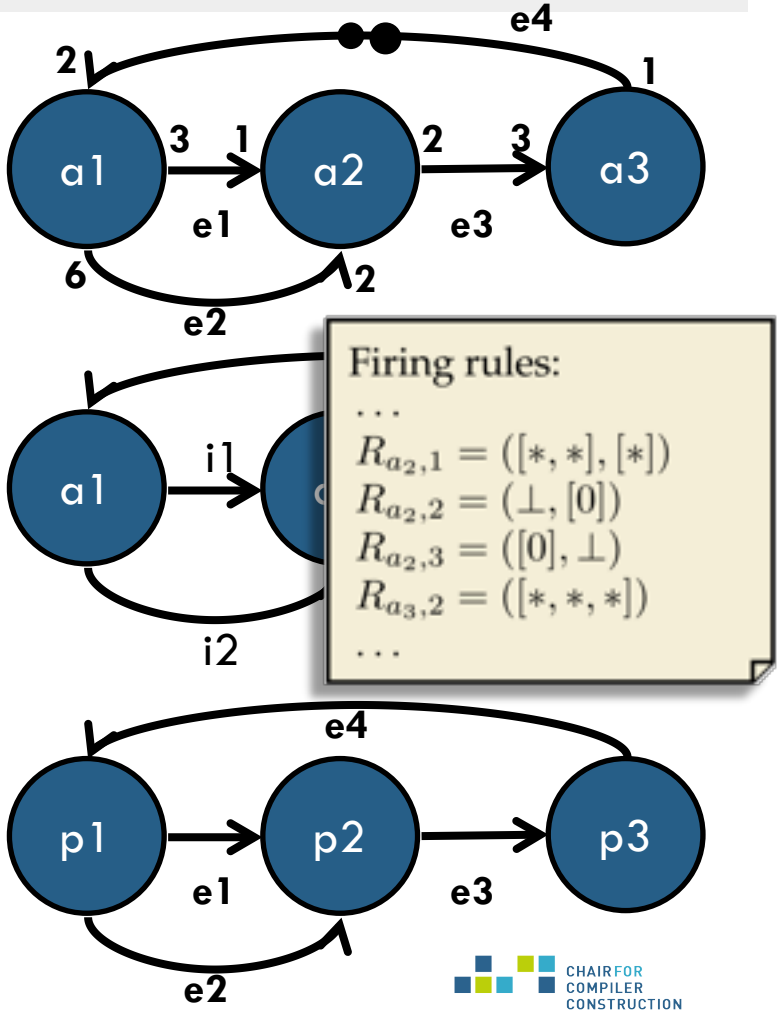
Architecture model




```
PNargs_ifft_r.ID = 6U;
    PNargs_ifft_r.PNchannel_freq_coef = filtered_coe
    PNargs_ifft_r.PNnum_freq_coef = 0U;
    PNargs_ifft_r.PNchannel_time_coef = sink_right;
    PNargs_ifft_r.channel = 1;
    sink_left = IPCllmrf_open(3, 1, 1);
    sink_right = IPCllmrf_open(7, 1, 1);
    PNargs_sink.ID = 7U;
    PNargs_sink.PNchannel_in_left = sink_left;
    PNargs_sink.PNnum_in_left = 0U;
    PNargs_sink.PNchannel_in_right = sink_right;
    PNargs_sink.PNnum_in_right = 0U;
    taskParams.arg0 = (xdc_UArg)&PNargs_src;
    taskParams.priority = 1;

ti_sysbios_knl_Task_create((ti_sysbios_knl_Task_Fun
&taskParams, &eb);
    glob_proc_cnt++;
    hasProcess = 1;
    taskParams.arg0 = (xdc_UArg)&PNargs_fft_1;
    taskParams.priority = 1;

ti_sysbios_knl_Task_create((ti_sysbios_knl_Task_Fun
ft_Templ, &taskParams, &eb);
    glob_proc_cnt++;
    hasProcess = 1;
    taskParams.arg0 = (xdc_UArg)&PNargs_ifft_r;
    taskParams.priority = 1;

ti_sysbios_knl_Task_create((ti_sysbios_knl_Task_Fun
fft_Templ, &taskParams, &eb);
    glob_proc_cnt++;
    hasProcess = 1;
    taskParams.arg0 = (xdc_UArg)&PNargs_sink;
    taskParams.priority = 1;
```

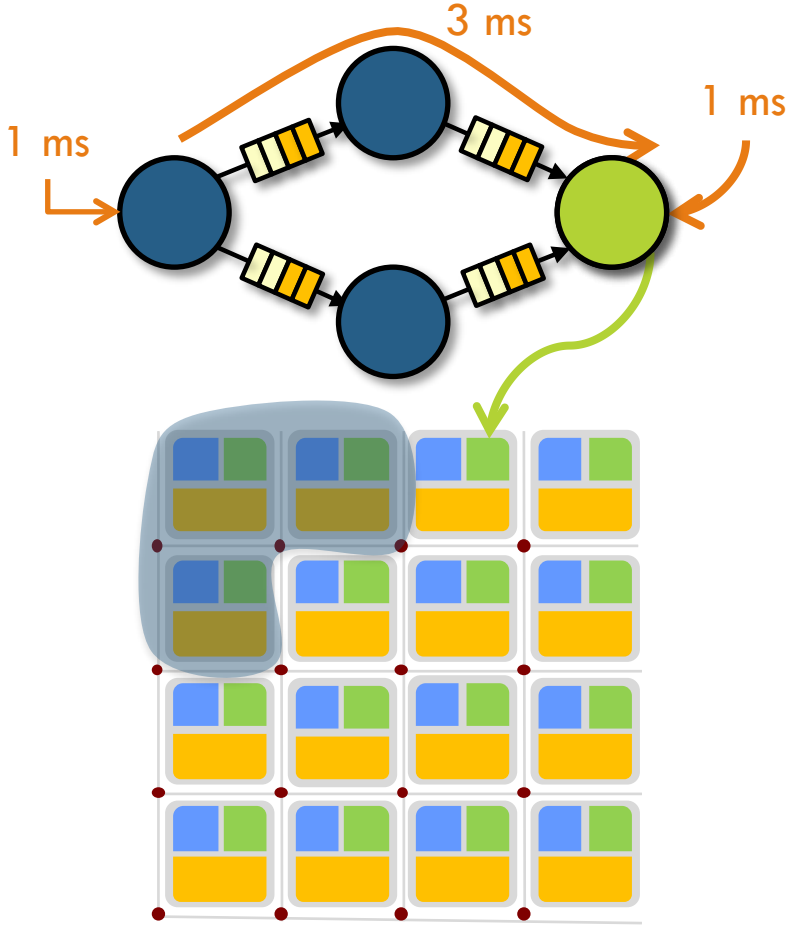
Application model: Beyond HSDF

- **SDF**: Popular model for *static* applications
  - Possible to derive equivalent HSDF
- Dynamic DF (**DDF**): For *dynamic* applications
  - Flexible firing rules (even non-determinism)
- Kahn Process Networks (**KPN**): Also dynamic
  - Deterministic
  - No firing semantics



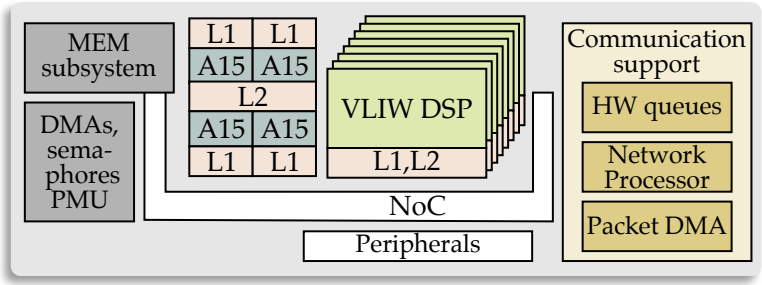
Non/extra-functional: Constraints

- Timing constraints
  - Process throughput
  - Latencies along paths
  - Time triggering
- Mapping constraints
  - Processes to processors
  - Channels to primitives
- Platform constraints
  - Subset of resources (processors or memories)
  - Utilization

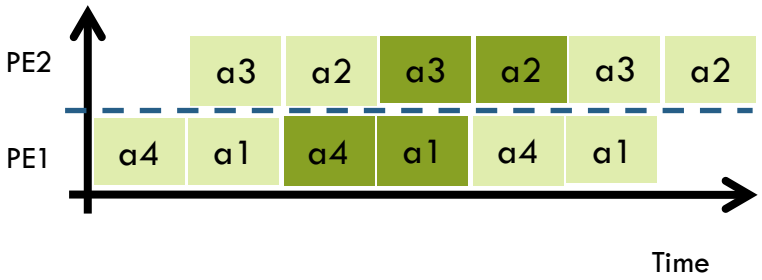


Architecture model

- Key for retargetable tool flows
  - Abstract for speed
  - Relevant components: processors (types), interconnect and memories



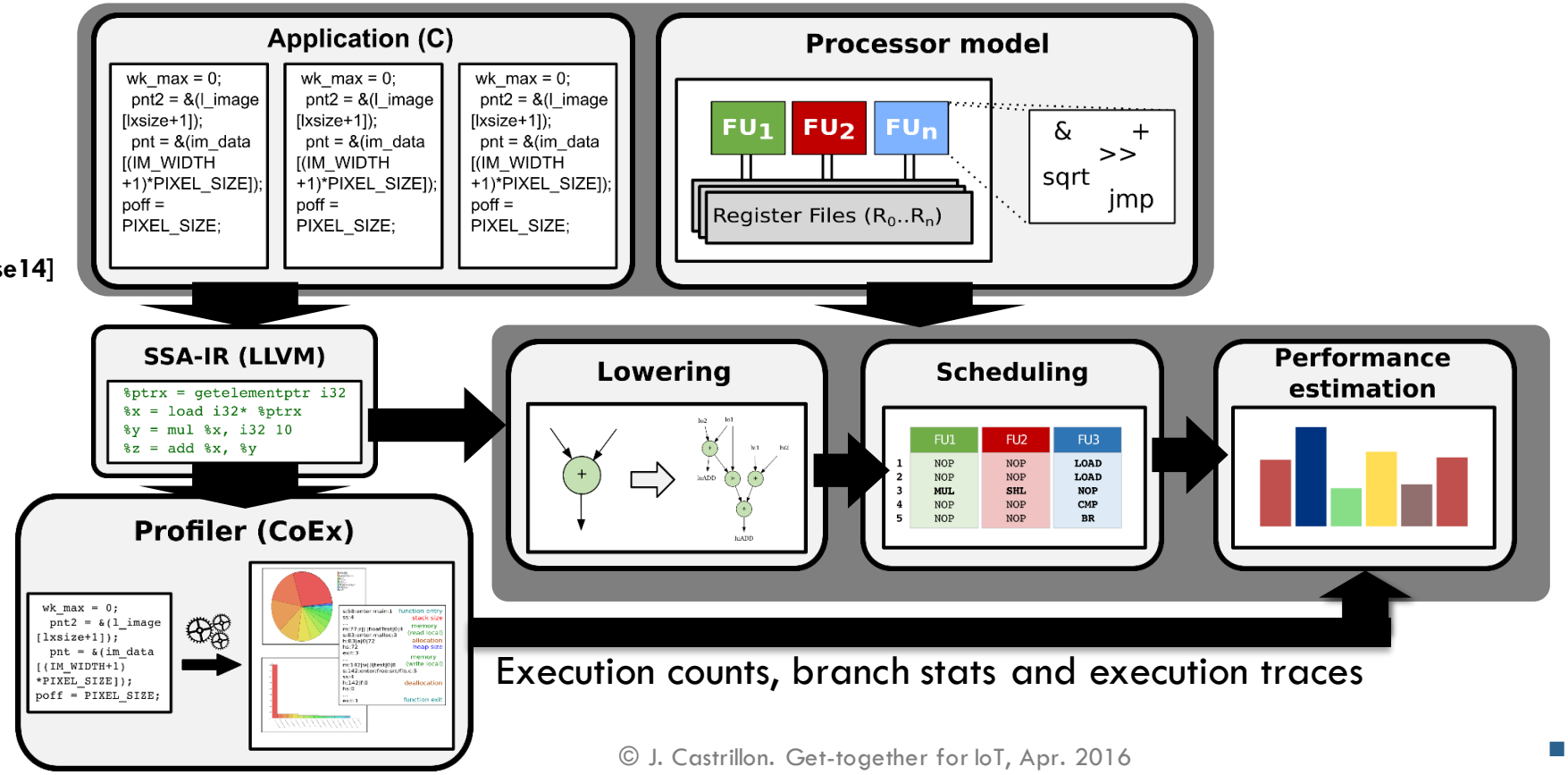
- Why? Understand effect of executing an actor on the different resources for a schedule



Modeling processors

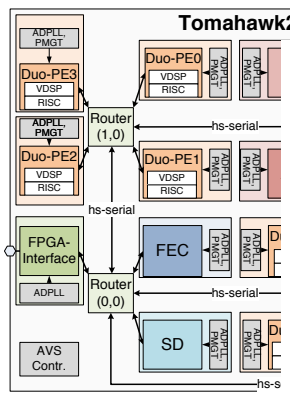
- Abstract models with functional units, latencies

[Euse14]

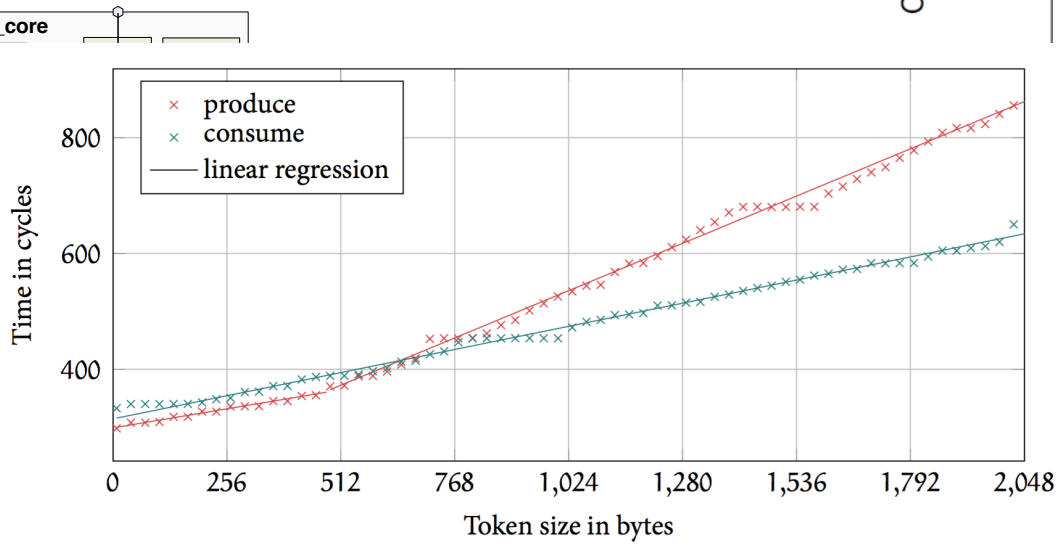


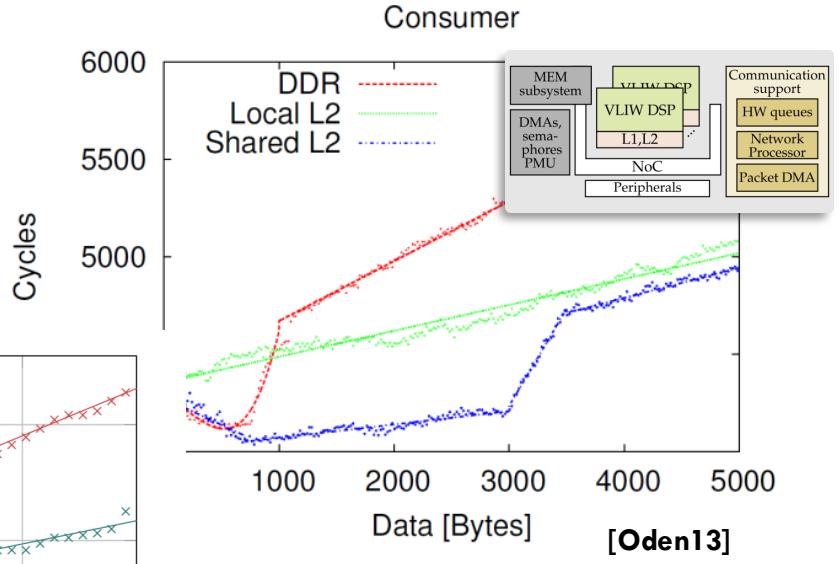
Modeling communication

- High-level: Available APIs for implementing application channels
- Benchmarking for different platforms



[Arnold13]

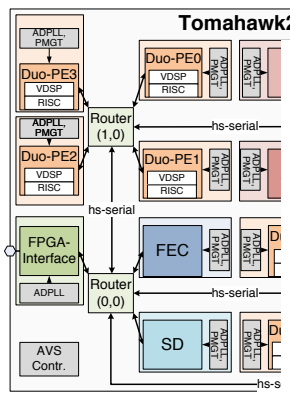




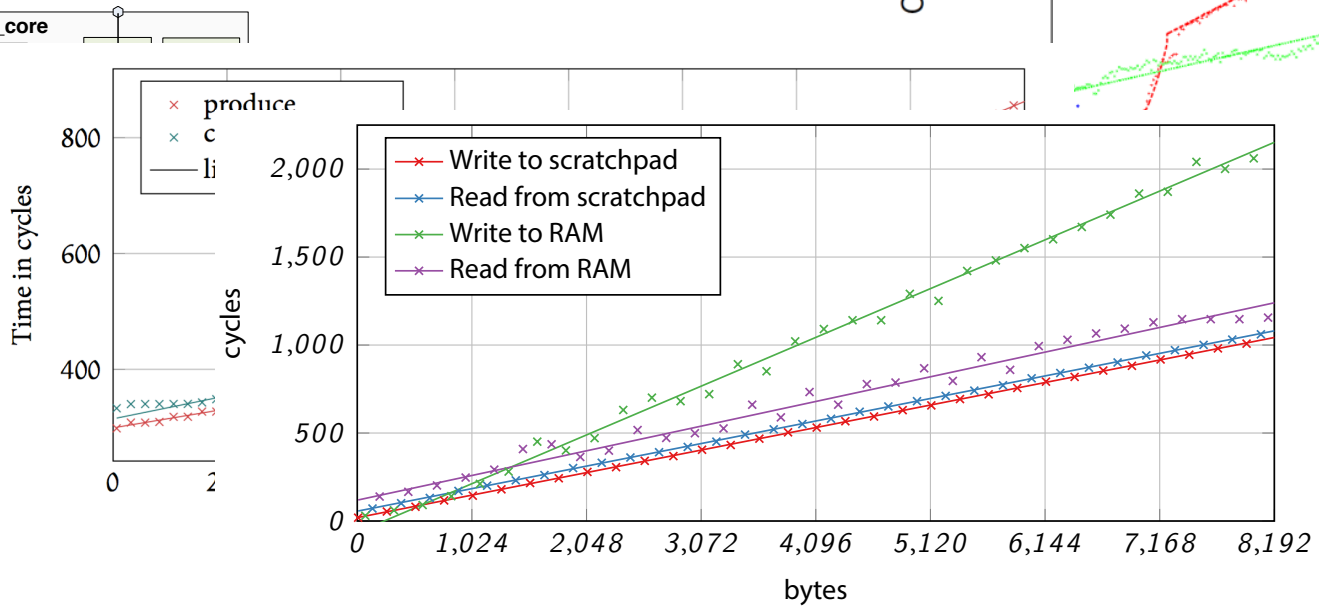
[Oden13]

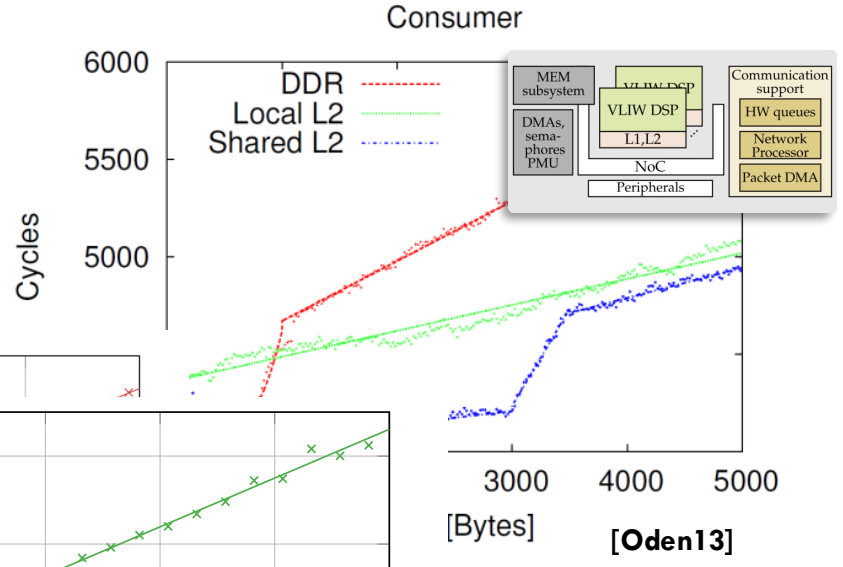
Modeling communication

- High-level: Available APIs for implementing application channels
- Benchmarking for different platforms



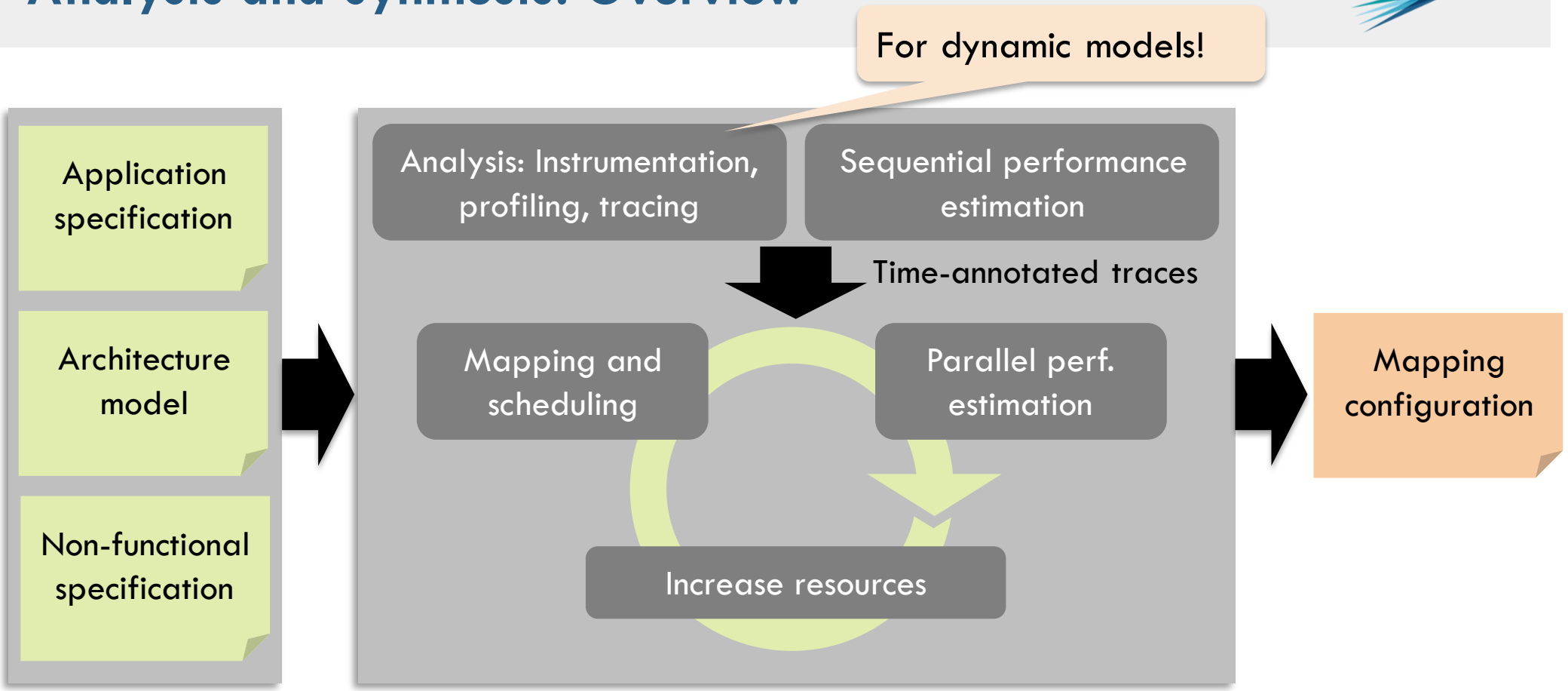
[Arnold13]





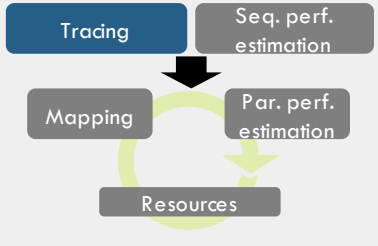
[Oden13]

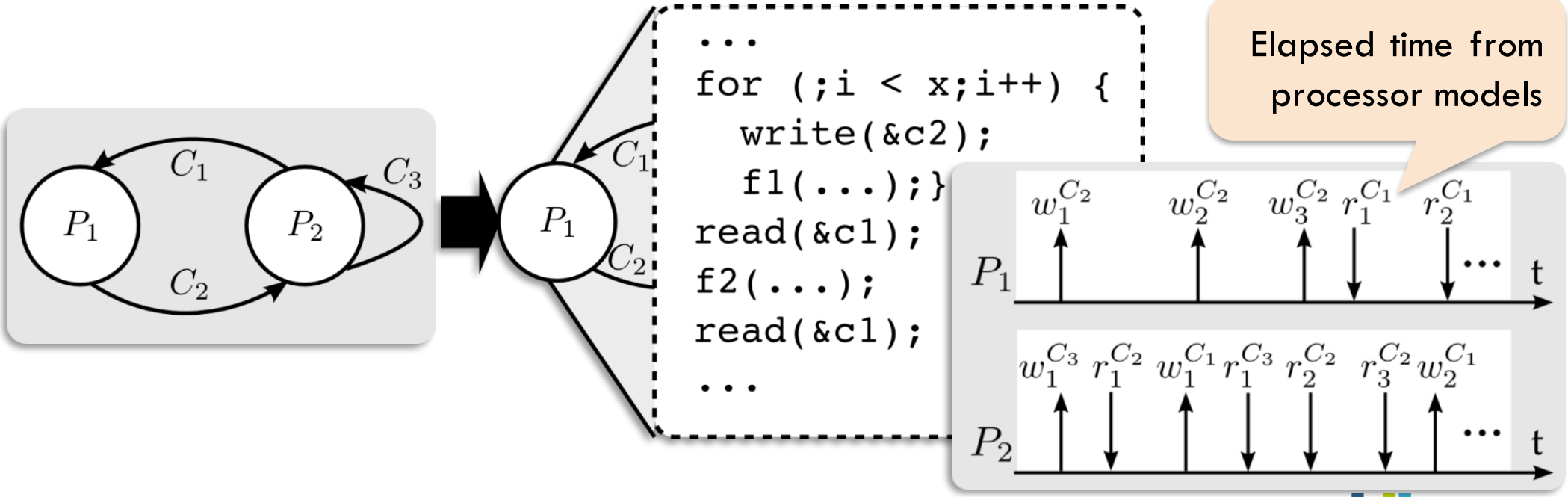
Analysis and synthesis: Overview



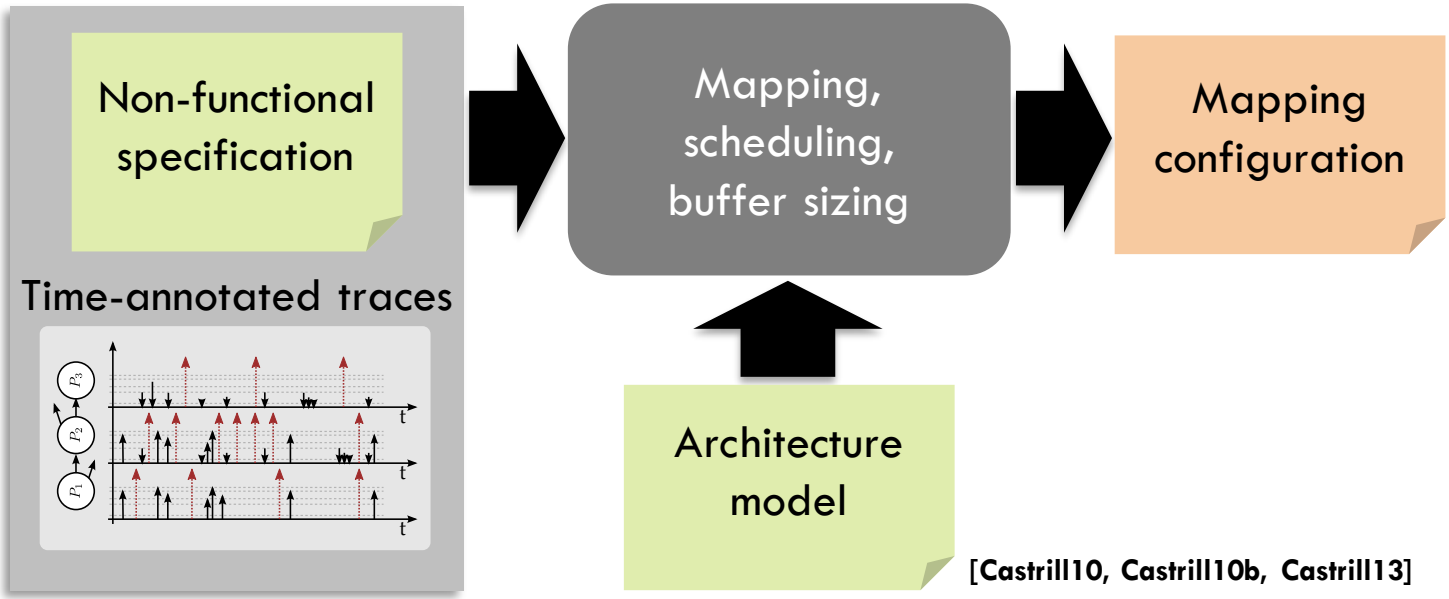
Tracing: Dealing with dynamic behavior

- KPNs do not have firing semantics
- **White model of processes**: source code analysis and tracing
- Tracing: instrumentation, token logging and event recording





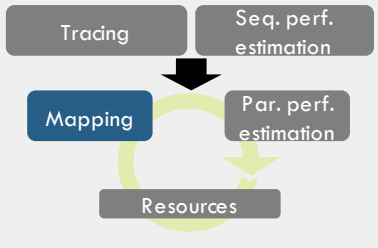
Trace-based synthesis



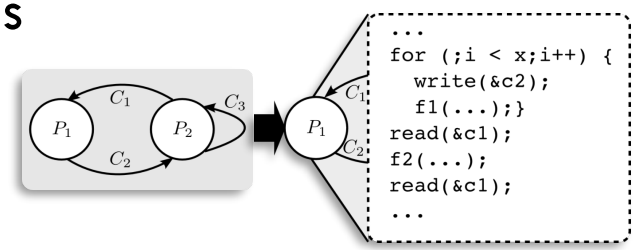
[Castrill10, Castrill10b, Castrill13]

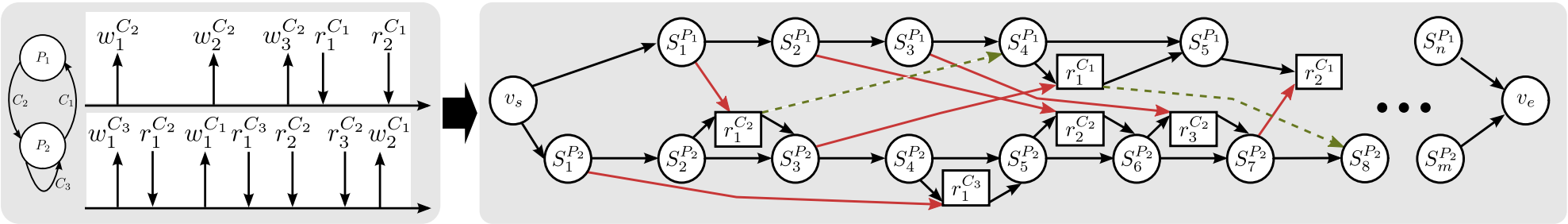
- Synthesis based on code and trace analysis (using simple heuristics)
  - Mapping of processes and channels
  - Scheduling policies
  - Buffer sizing

Mapping, scheduling and buffer sizing



- Graph representation of event traces
  - Reason about deadlocks
  - Find critical path

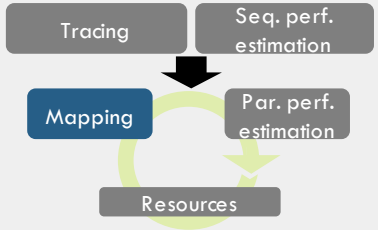




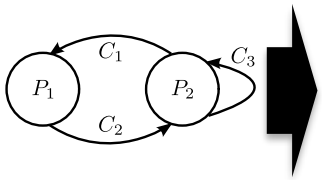
[Castrill12]

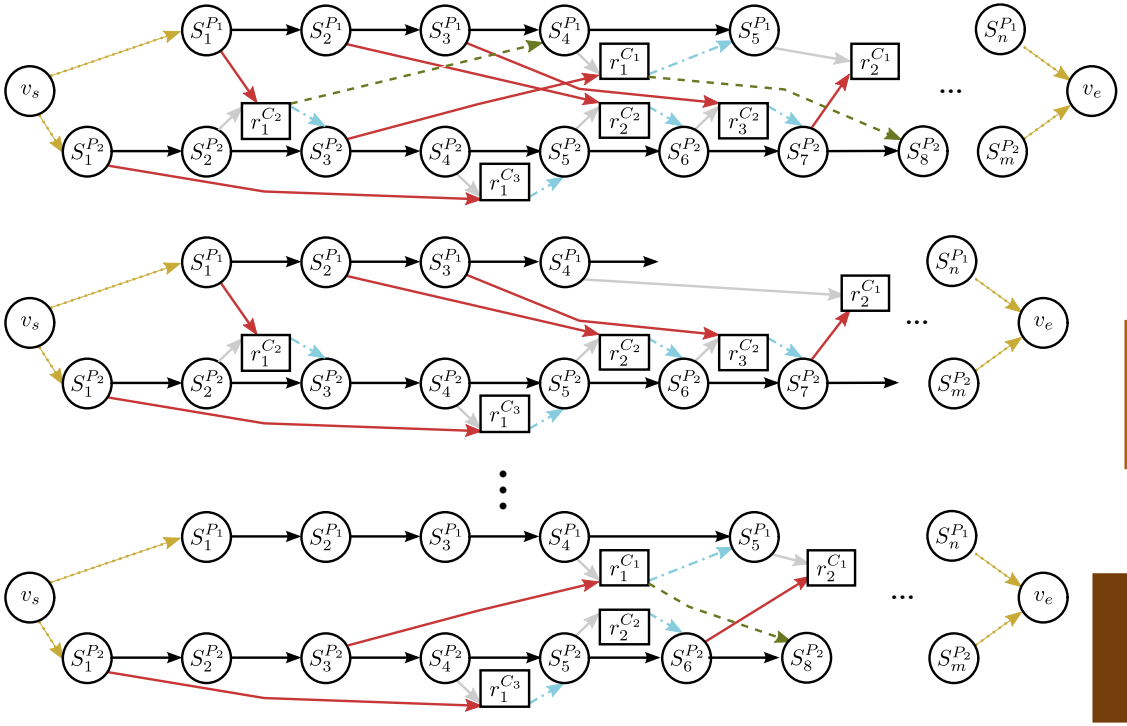
- Downside: For dynamic portions of the application, graph is input dependent

Multiple traces (ongoing work)



- Different input → different behavior (traces)
  - Characterize behaviors and impact on mapping performance





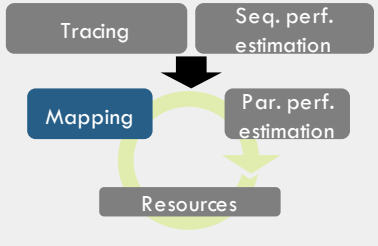
Mapping configuration

Mapping configuration

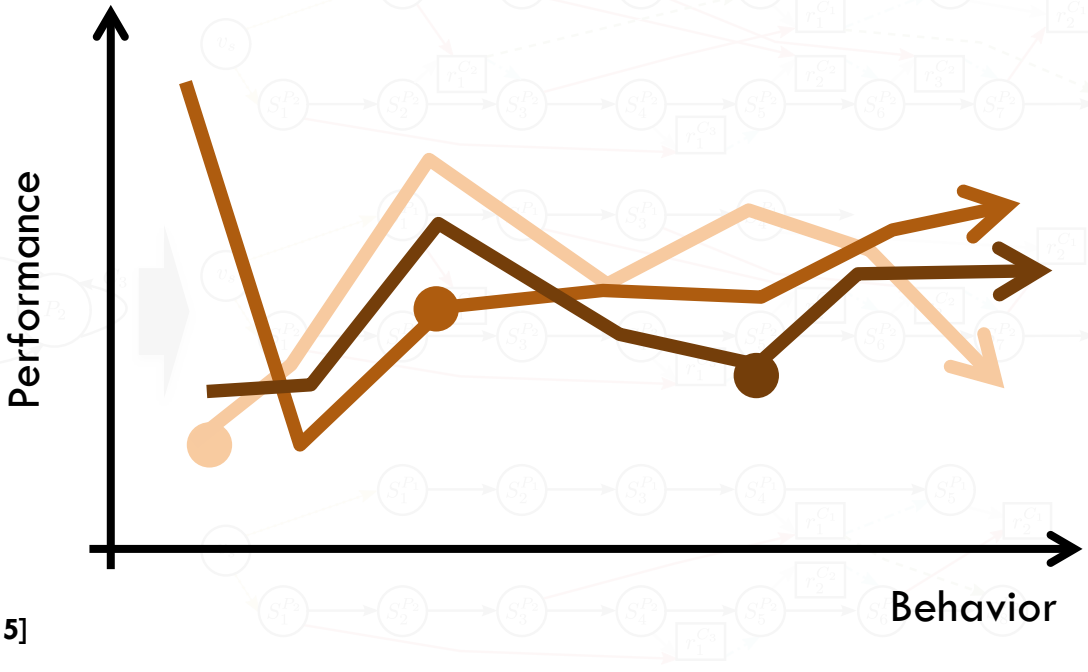
Mapping configuration

[Goens15]

Multiple traces (2)



- Different input → different behavior (traces)
  - Characterize behaviors and impact on mapping performance



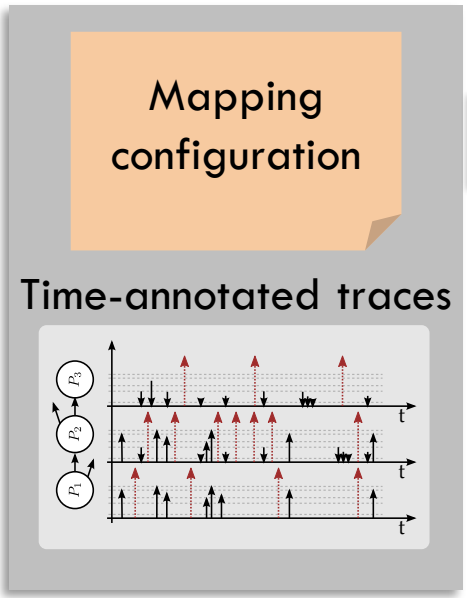
Mapping configuration

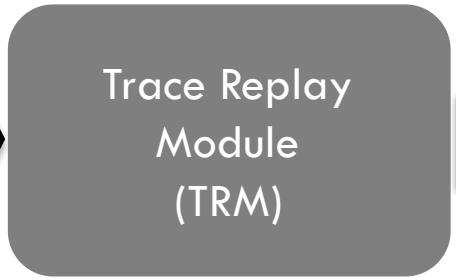
Mapping configuration

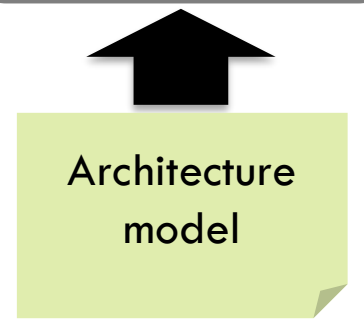
Mapping configuration

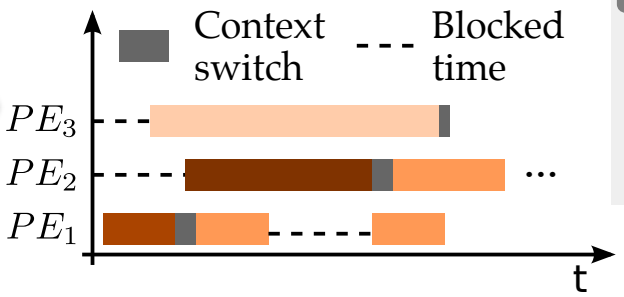
[Goens15]

Parallel performance estimation

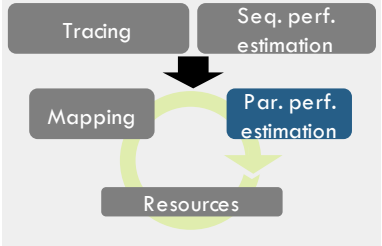






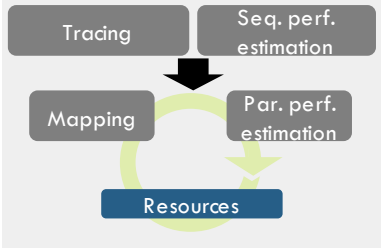


Grantt Chart, platform utilization, channel profiles, ...

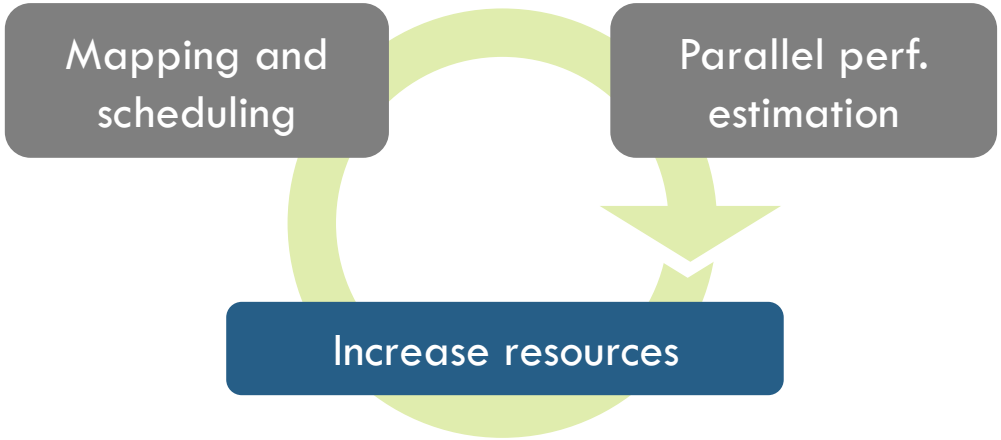


- Discrete event simulator to evaluate a solution
  - Replay traces according to mapping
  - Extract costs from architecture file (NoC modeling, context switches, communication)

Increasing resources



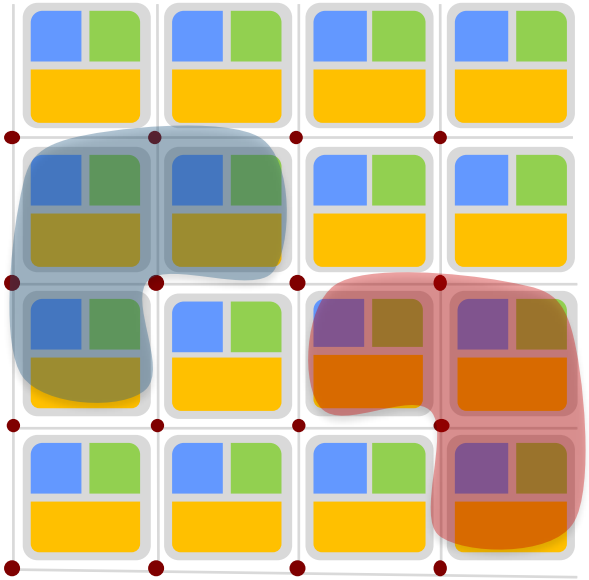
- Add resources to the synthesis until constraints are met

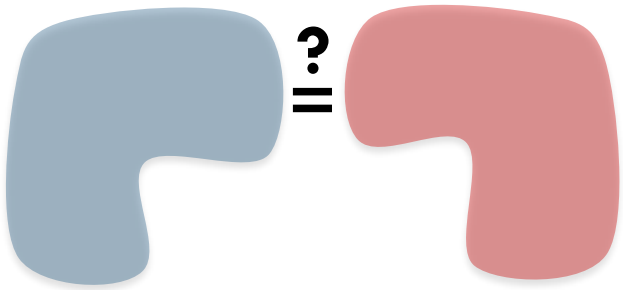


- Add processors and memories
  - Easy for homogeneous platforms
  - Non trivial for heterogeneous platforms (ongoing work)

Multiple mapping configurations

- In case multiple applications may run on the system (or system of systems)
  - Need different configurations
  - Adapt to resource and energy budgets
- Reason about mapping equivalences

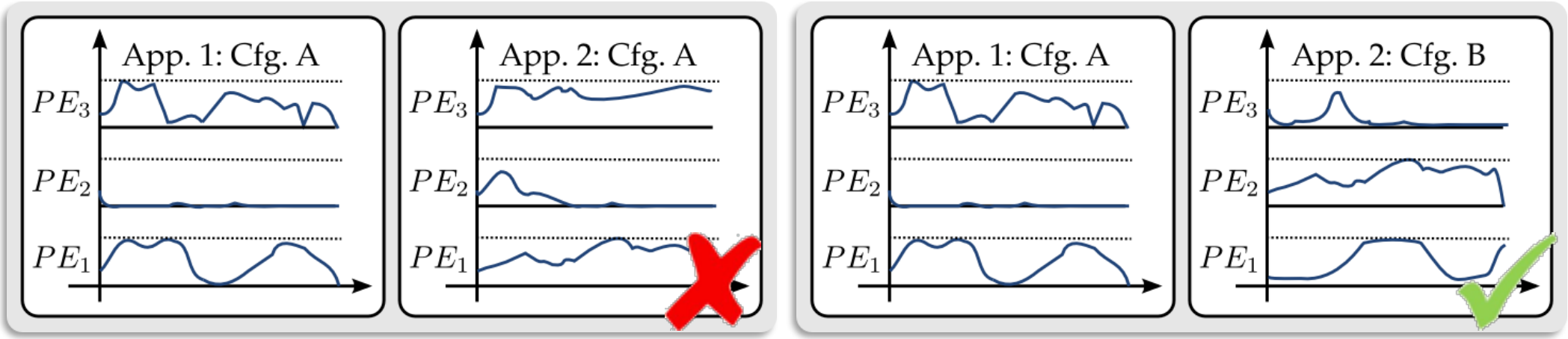




[Goens15]

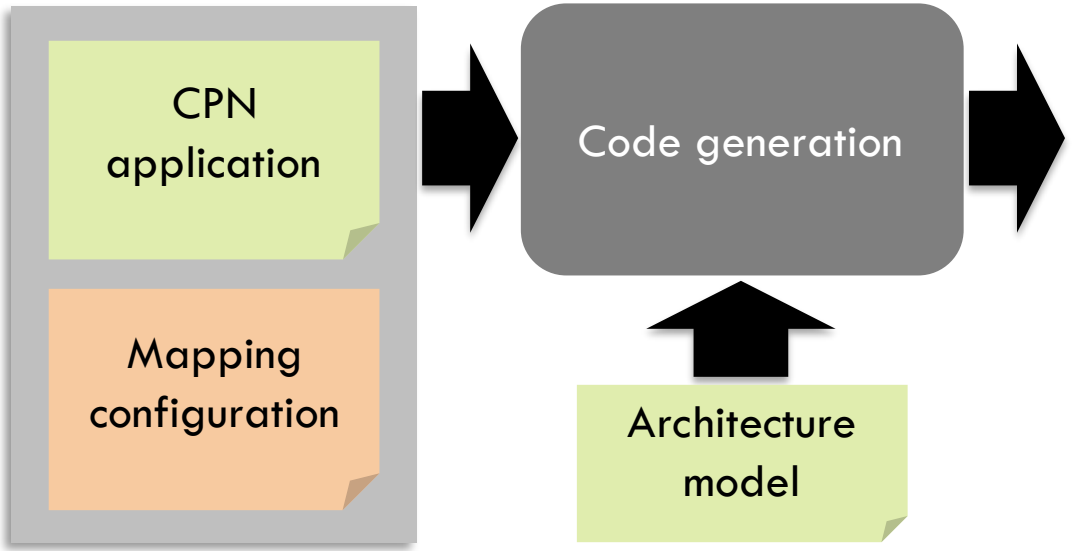
Analysis for Multi-applications

- Use utilization profiles from the discrete event simulator (TRM)
- Quickly separate good from bad implementations



[Castrill14]

Code generation




```
PNargs_ifft_r.ID = 6U;
   PNargs_ifft_r.PNchannel_freq_coef = filtered_coef_right;
   PNargs_ifft_r.PNnum_freq_coef = 0U;
   PNargs_ifft_r.PNchannel_time_coef = sink_right;
   PNargs_ifft_r.channel = 1;
   sink_left = IPCllmrf_open(3, 1, 1);
   sink_right = IPCllmrf_open(7, 1, 1);
   PNargs_sink.ID = 7U;
   PNargs_sink.PNchannel_in_left = sink_left;
   PNargs_sink.PNnum_in_left = 0U;
   PNargs_sink.PNchannel_in_right = sink_right;
   PNargs_sink.PNnum_in_right = 0U;
   taskParams.arg0 = (xdc_UArg)&PNargs_src;
   taskParams.priority = 1;
   ...
```

Debugging info

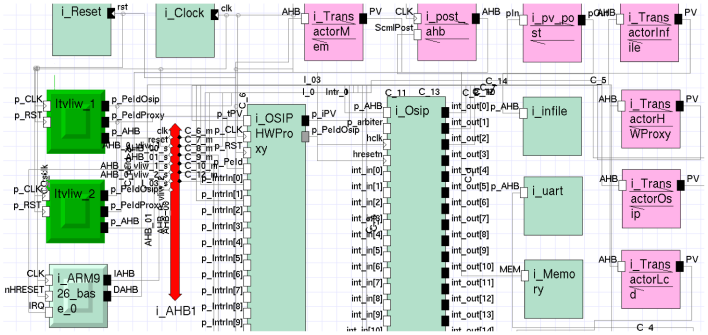
[Castrill11b, Murillo14]

- Take mapping configuration and generate code accordingly
- From architecture model: APIs, configuration parameters, …
- Sample targets: experimental heterogenous systems, TI (Keysonte, TDA3x), Parallella/Ephiphany, ARM-based (Exynos, Snapdragon)

SILEXICA

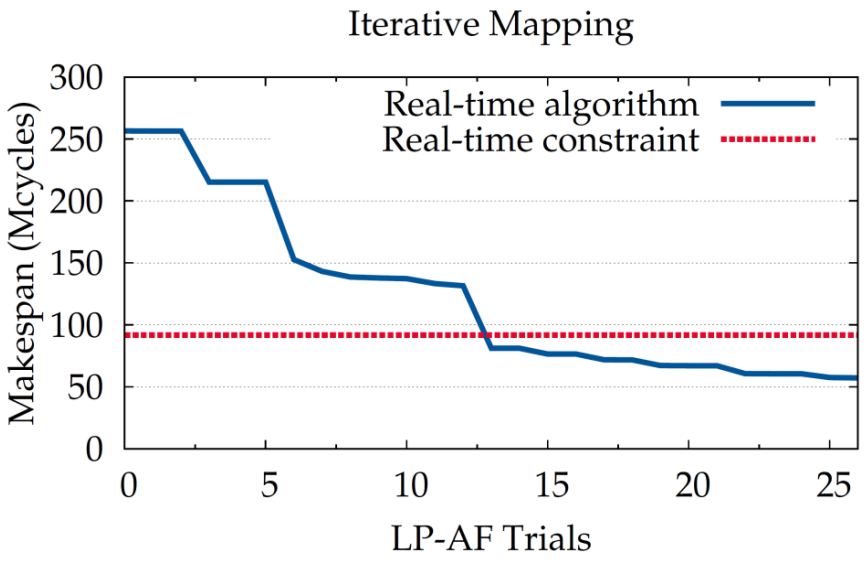
Examples

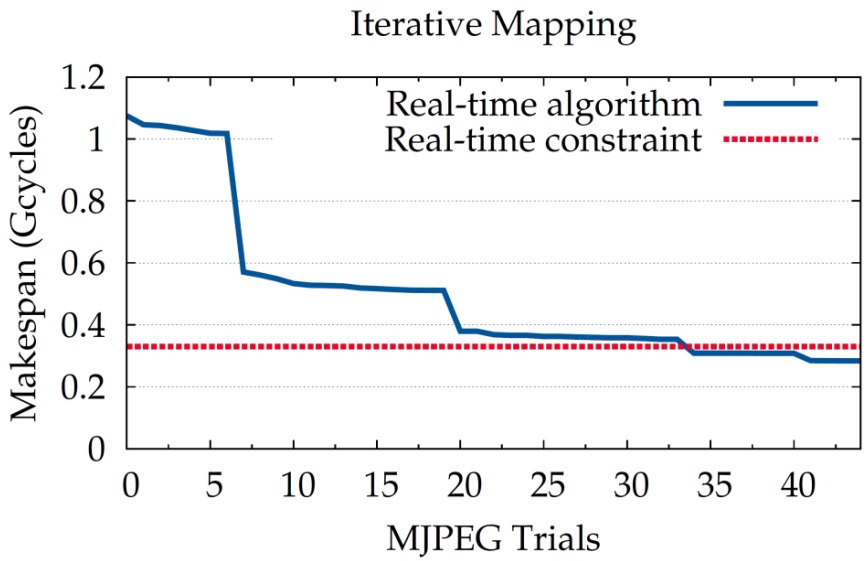
- Same application(s) automatically compiled to different platforms
- Virtual platforms: SystemC models of full systems
  - Bare metal
  - Multi-RISC/VLIW platform
- Real platform: TI Keystone platform
  - Speedup on commercial platforms
  - Code generation against vendor stacks



Example: multi-media applications

- Iterative mapping





Manual vs. Automatic: TI Keystone

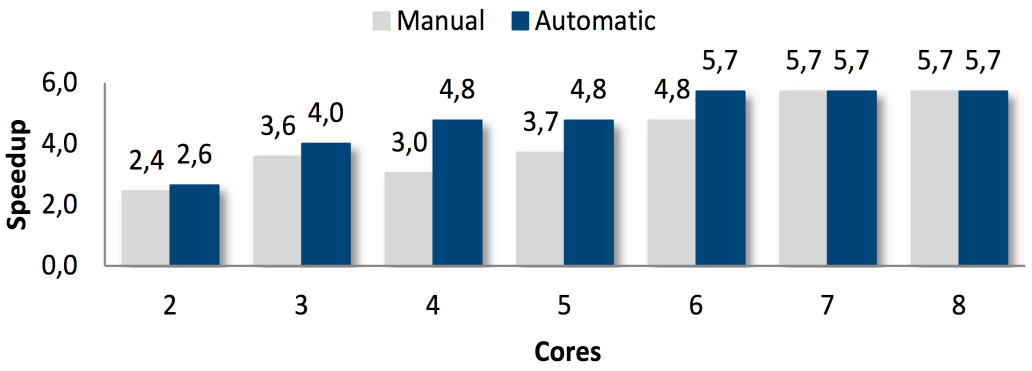
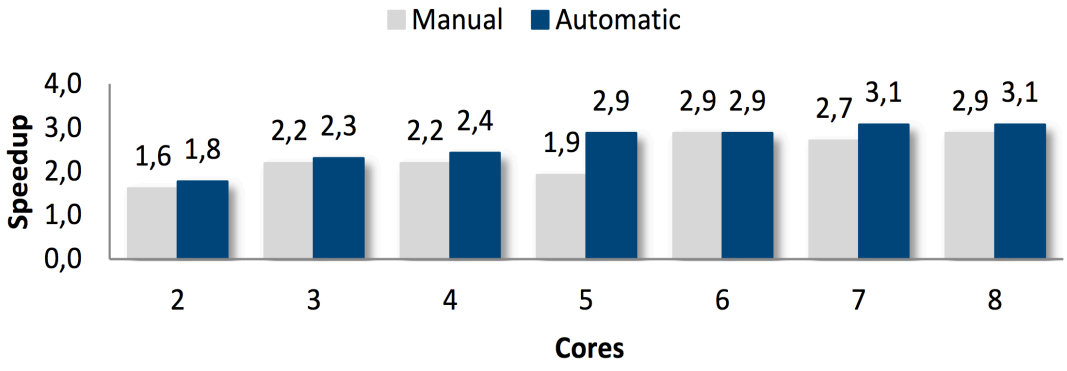
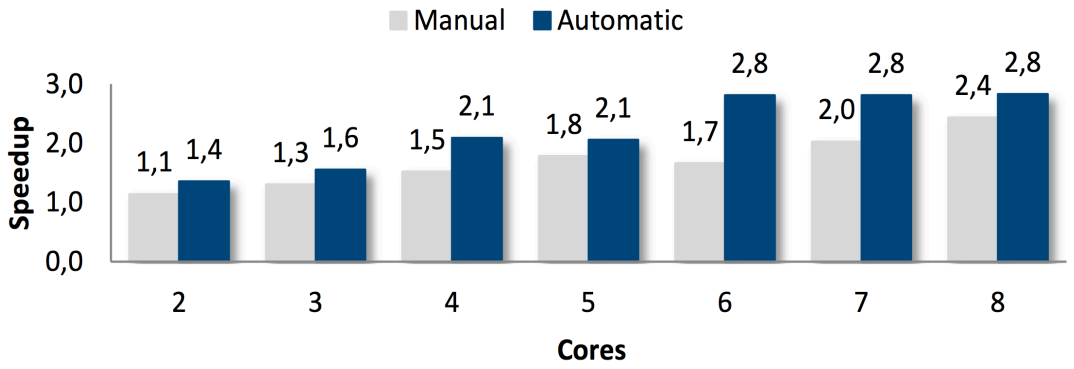


Image processing



Audio filtering application



LTE digital receiver

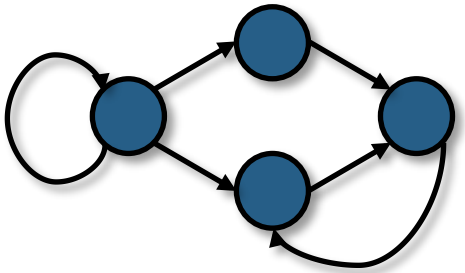
[Aguilar14]

Outline

- Introduction
- Models of computation
- Tool flows
- **Outlook for IoT**
- Summary

So far…

MoC-based Application

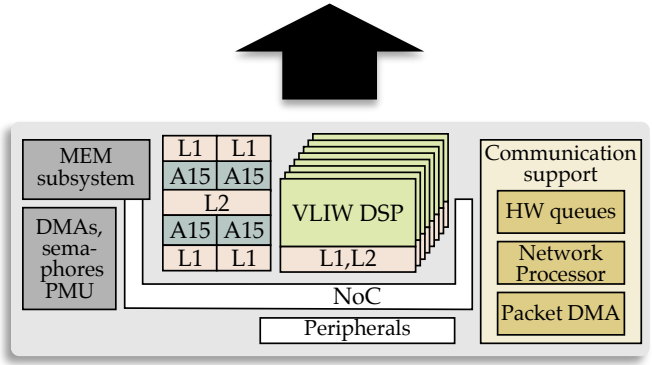


Non-functional specification

Analysis

Synthesis

Code generation

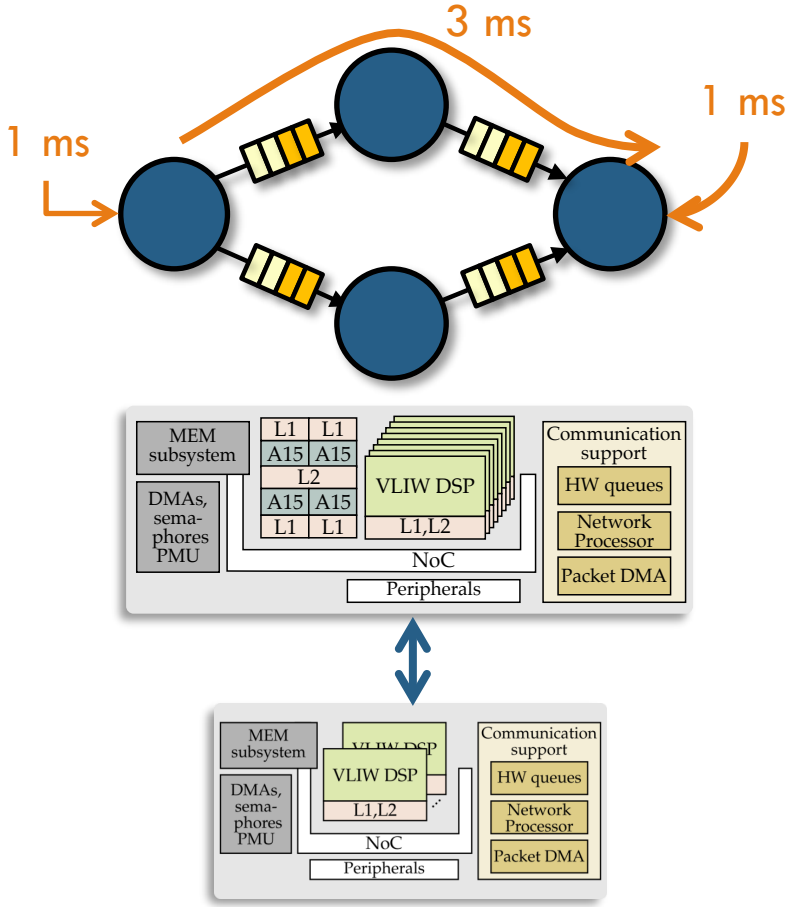


Architecture model

```
PNargs_ifft_r.ID = 6U;
    PNargs_ifft_r.PNchannel_freq_coef = filtered_coe
    PNargs_ifft_r.PNnum_freq_coef = 0U;
    PNargs_ifft_r.PNchannel_time_coef = sink_right;
    PNargs_ifft_r.channel = 1;
    sink_left = IPCllmrf_open(3, 1, 1);
    sink_right = IPCllmrf_open(7, 1, 1);
    PNargs_sink.ID = 7U;
    PNargs_sink.PNchannel_in_left = sink_left;
    PNargs_sink.PNnum_in_left = 0U;
    PNargs_sink.PNchannel_in_right = sink_right;
    PNargs_sink.PNnum_in_right = 0U;
    taskParams.arg0 = (xdc_UArg)&PNargs_src;
    taskParams.priority = 1;

ti_sysbios_knl_Task_create((ti_sysbios_knl_Task_Fun
&taskParams, &eb);
    glob_proc_cnt++;
    hasProcess = 1;
    taskParams.arg0 = (xdc_UArg)&PNargs_fft_1;
    taskParams.priority = 1;

ti_sysbios_knl_Task_create((ti_sysbios_knl_Task_Fun
ft_Templ, &taskParams, &eb);
    glob_proc_cnt++;
    hasProcess = 1;
    taskParams.arg0 = (xdc_UArg)&PNargs_ifft_r;
    taskParams.priority = 1;

ti_sysbios_knl_Task_create((ti_sysbios_knl_Task_Fun
fft_Templ, &taskParams, &eb);
    glob_proc_cnt++;
    hasProcess = 1;
    taskParams.arg0 = (xdc_UArg)&PNargs_sink;
    taskParams.priority = 1;
```

What's missing?

- Reactivity
  - Supported for periodic events (trigger)
  - Enough for many applications (automotive)
  - Sporadic (I/O) events may break determinism
- Better execution guarantees
- Multi-SoC
  - Supported if inter-SoC communication is predictable (same cost-function approach)
  - Scalability: Add a layer of "geographic" coarse mapping



Outline

- Introduction
- Models of computation
- Tool flows
- Outlook for IoT
- **Summary**

Summary

- IoT complexity: heterogeneity, resource constraints, distributed, …
- The need for formal MoC for programming and synthesis
- Sample tool flow
  - Heterogeneity: High-level architecture models
  - Support dynamic applications: Analysis and synthesis based on traces
  - Extensions for: multiple traces
- Outlook
  - Adaptivity at runtime
  - Reactivity and multi-SoC case studies

Acknowledgements

- Silexica Software Solutions GmbH

- German Cluster of Excellence: Center for Advancing Electronics Dresden (www.cfaed.tu-dresden.de)

References

[**Castrill14**] J. Castrillon , et al., "Programming Heterogeneous MPSoCs: Tool Flows to Close the Software Productivity Gap", Springer, 2014

[**Castrill11a**] J. Castrillon, et al., "Trends in embedded software synthesis", International Conference on Embedded Computer Systems (SAMOS) pp. 347-354, 2011

[**Oden13**] M. Odendahl, et al., "Split-cost communication model for improved MPSoC application mapping", In International Symposium on System on Chip pp. 1-8, 2013

[**Arnold13**] O. Arnold, et al. "Tomahawk - Parallelism and Heterogeneity in Communications Signal Processing MPSoCs". *TECS, 2013*

[**Eusse14**] J.F. Eusse, , et al., "Pre-architectural performance estimation for ASIP design based on abstract processor models," In SAMOS 2014 pp.133-140, 2014

[**Castrill10**] J. Castrillon, , et al., "Component-based waveform development: The nucleus tool flow for efficient and portable SDR," Wireless Innovation Conference and Product Exposition (SDR), 2010

[**Castrill10b**] J. Castrillon, et al., "Trace-based KPN composability analysis for mapping simultaneous applications to MPSoC platforms", In DATE 2010, pp. 753-758

[**Castrill13**] J. Castrillon, et al., "MAPS: Mapping concurrent dataflow applications to heterogeneous MPSoCs," IEEE Transactions on Industrial Informatics, vol. 9, 2013

[**Castrill12**] J. Castrillon, et al. , "Communication-aware mapping of KPN applications onto heterogeneous MPSoCs," in DAC 2012

[**Goens15**] A. Goens, et al., "Analysis of Process Traces for Mapping Dynamic KPN Applications to MPSoCs", In IFIP International Embedded Systems Symposium (IESS), 2015, Foz do Iguaçu, Brazil (Accepted for publication), 2015

[**Castrill11b**] J. Castrillon, et al., "Backend for virtual platforms with hardware scheduler in the MAPS framework", In LASCAS 2011 pp. 1-4, 2011

[**Murillo14**] L. G. Murillo, et al., "Automatic detection of concurrency bugs through event ordering constraints", In DATE 2014, pp. 1-6, 2014

[**Aguilar14**] M. Aguilar, et al., "Improving performance and productivity for software development on TI Multicore DSP platforms," in Education and Research Conference (EDERC), 2014 6th European Embedded Design in , vol., no., pp.31-35, 11-12 Sept. 2014