CHAIR FOR COMPILER CONSTRUCTION

cfaed CENTER FOR ADVANCING ELECTRONICS DRESDEN

# Programming for adaptive and energy-efficient computing

Jeronimo Castrillon

Chair for Compiler Construction (CCC)

TU Dresden, Germany

International Conference on High Performance Compilation, Computing and Communications (HP3C)

March 23, 2017. Malaysia

# TU Dresden – Department of Computer Science

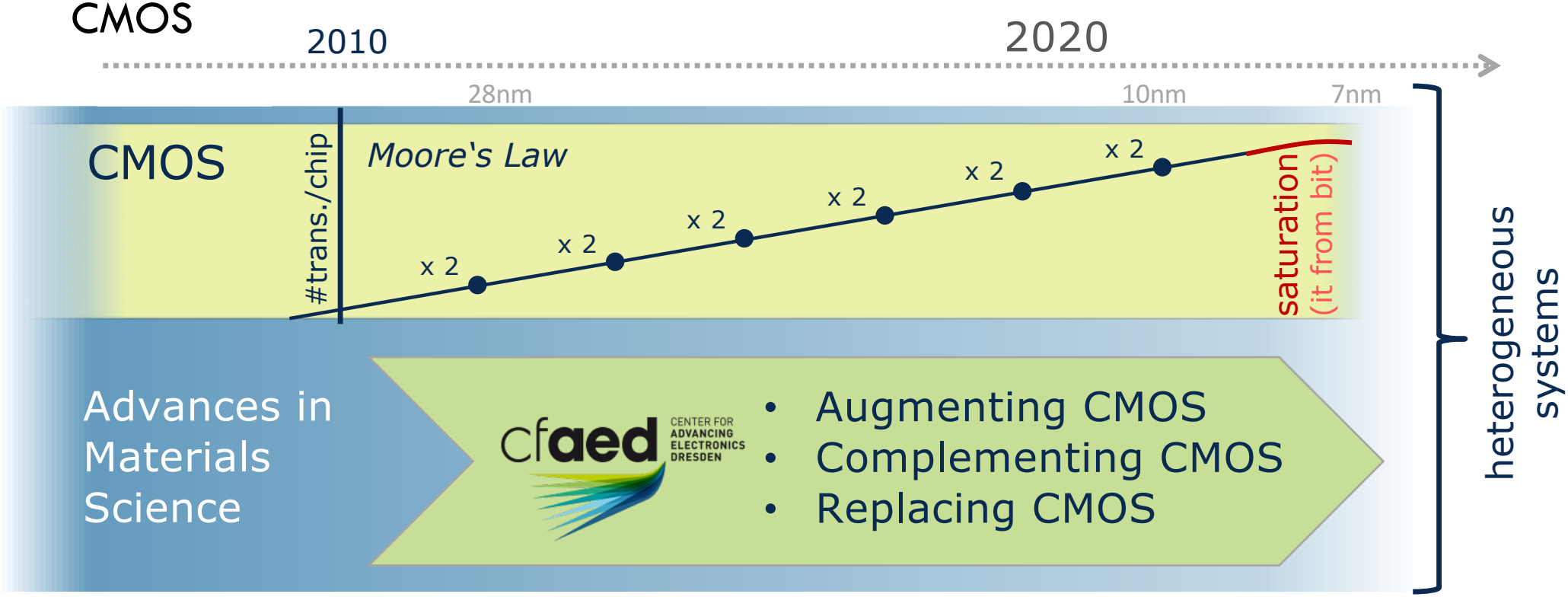


Est. 1828, one of Germany's **TU9**

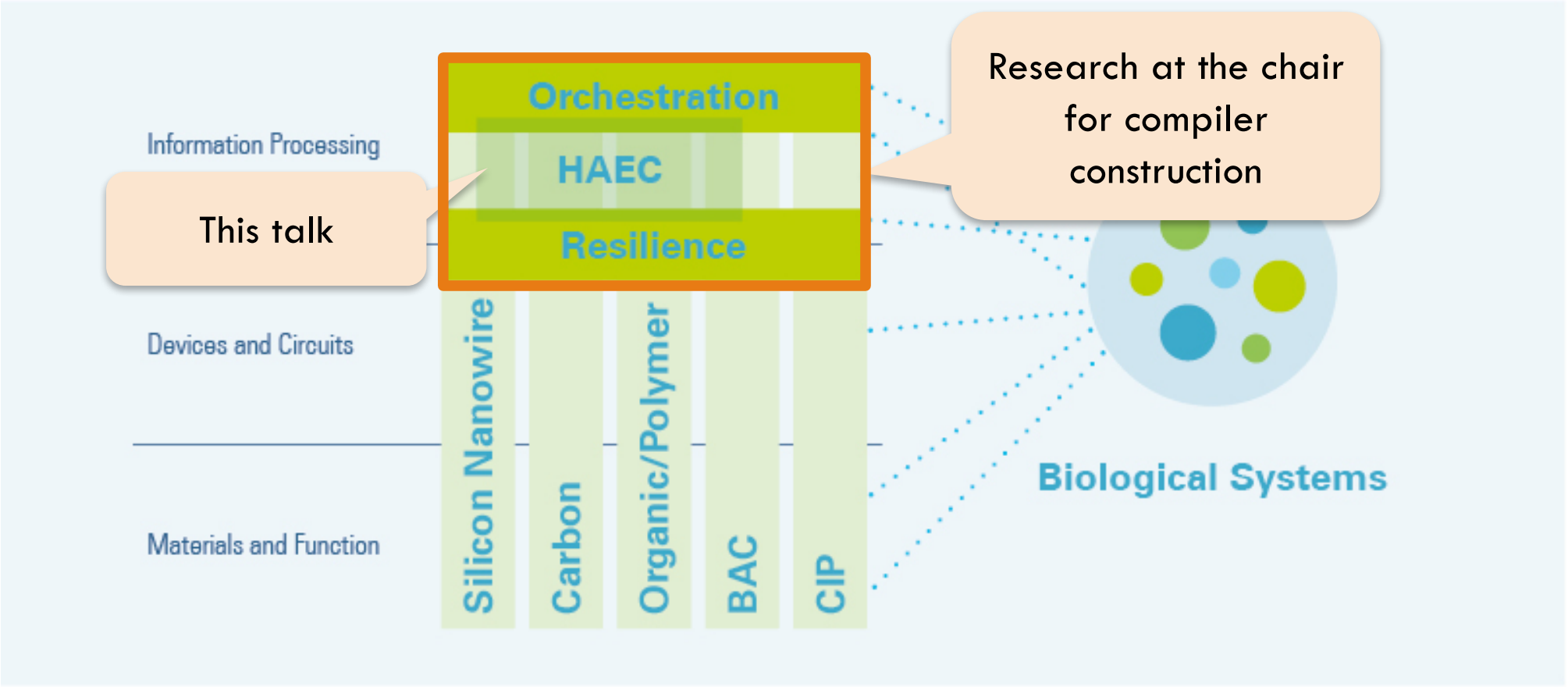
Est. 1900, 26 professorships

# Cfaed: Center for Advancing Electronics Dresden

- Explore new technologies for information processing that overcome the limits of CMOS



# cfaed: Structure



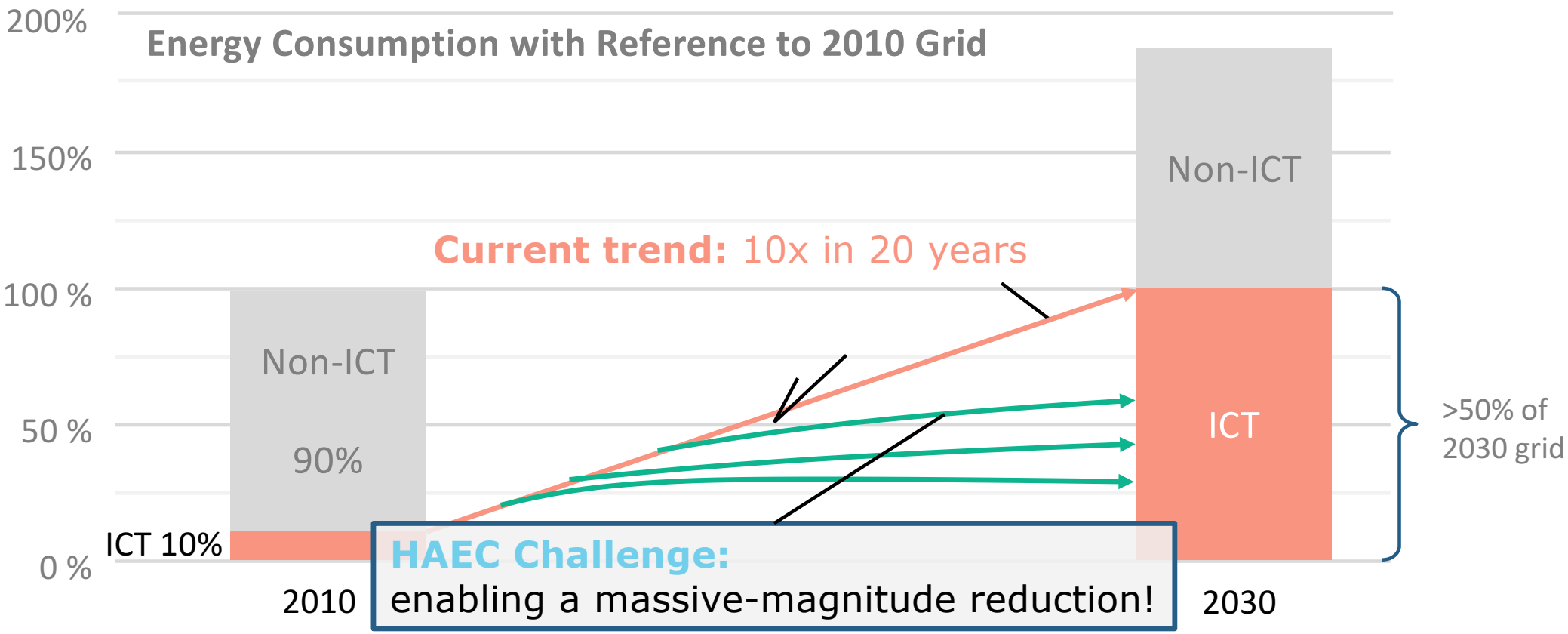
# Agenda

- Contextualization
- HAEC overview
- Dataflow programming
- Flexible mappings
- Closing remarks

# Agenda

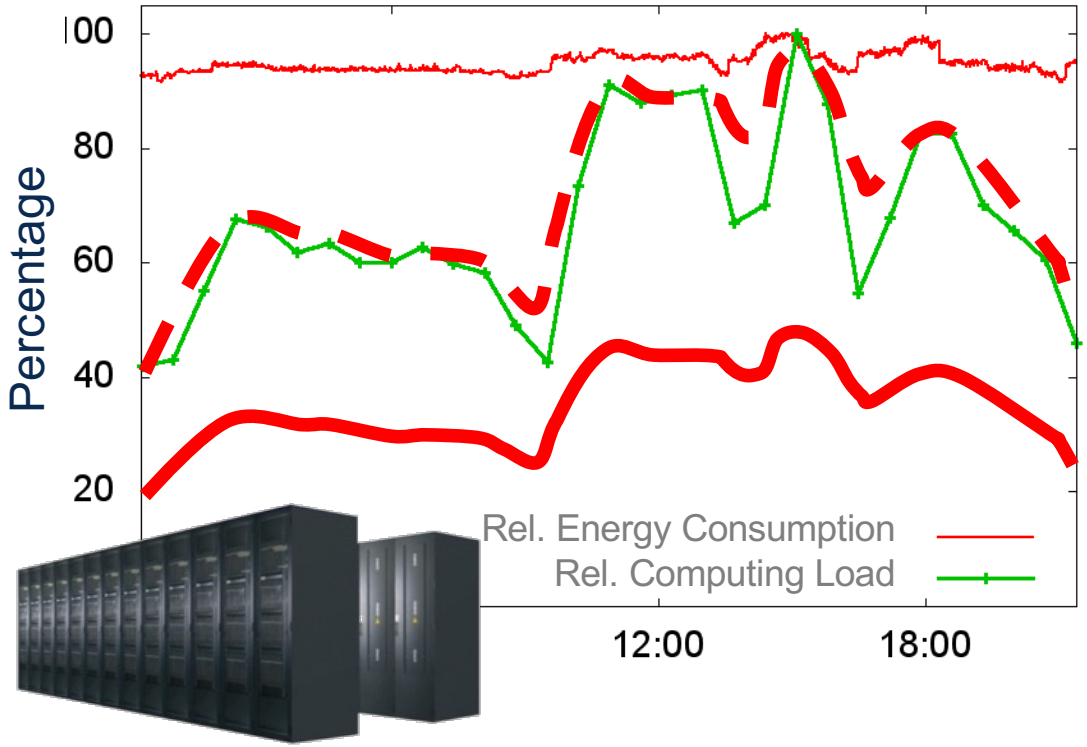
- Contextualization
- **HAEC overview**
- Dataflow programming
- Flexible mappings
- Closing remarks

# HAEC: Motivation



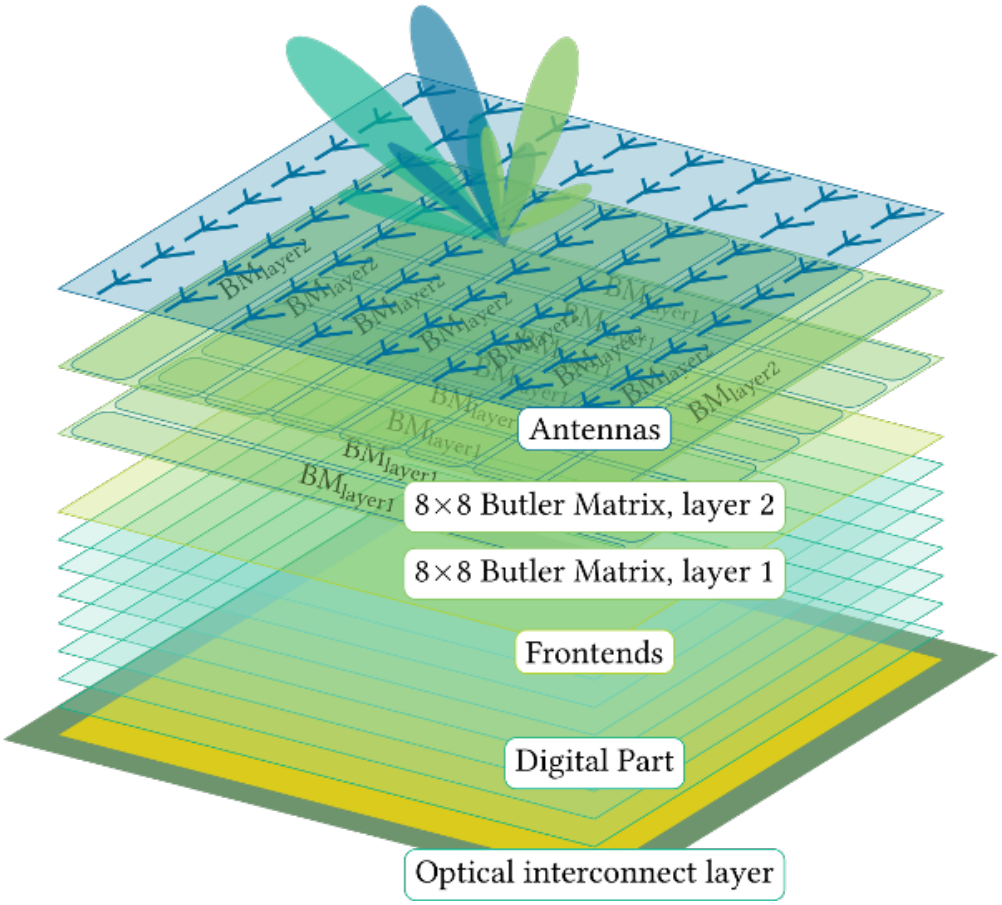
# HAEC: Highly-adaptive energy-efficient computing

Center for Information Services and High Performance Computing (ZIH)
Measurement at June 20, 2008

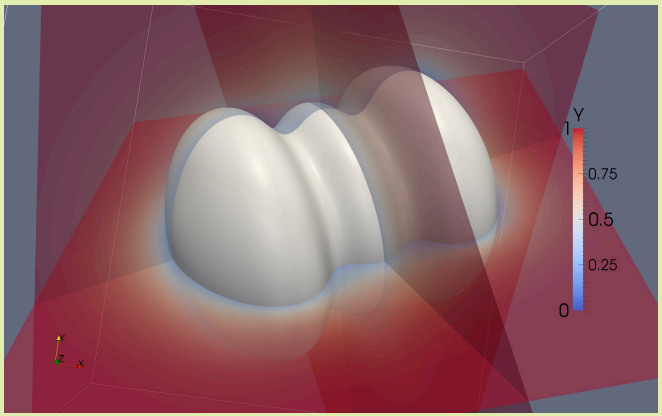



**Goal:**
Minimize energy via multi-layer SW/HW adaptivity

# The HAEC Box



# HAEC: Project structure




Courtesy: Prof. Fröhlich, TU Dresden

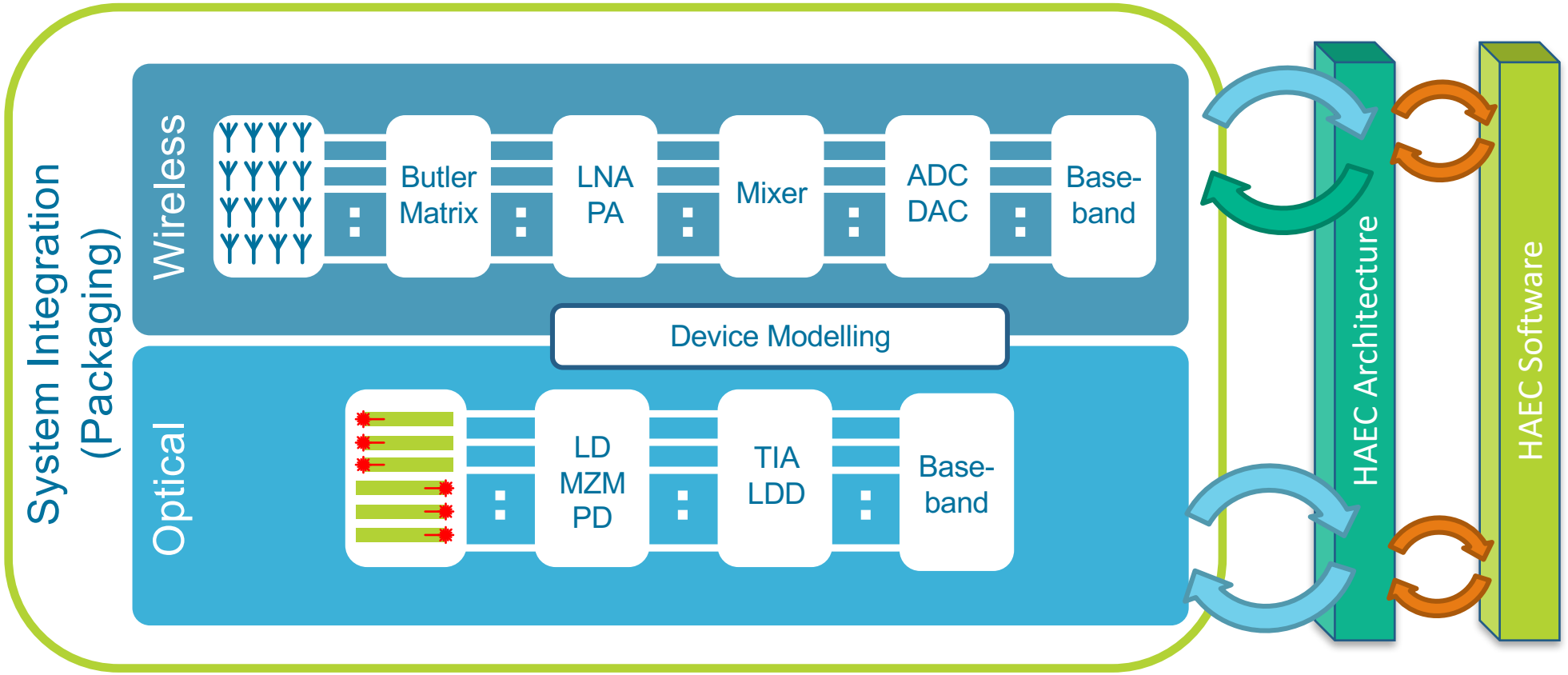
Adaptable Sofware

**HAEC-SW**

Flexible hardware

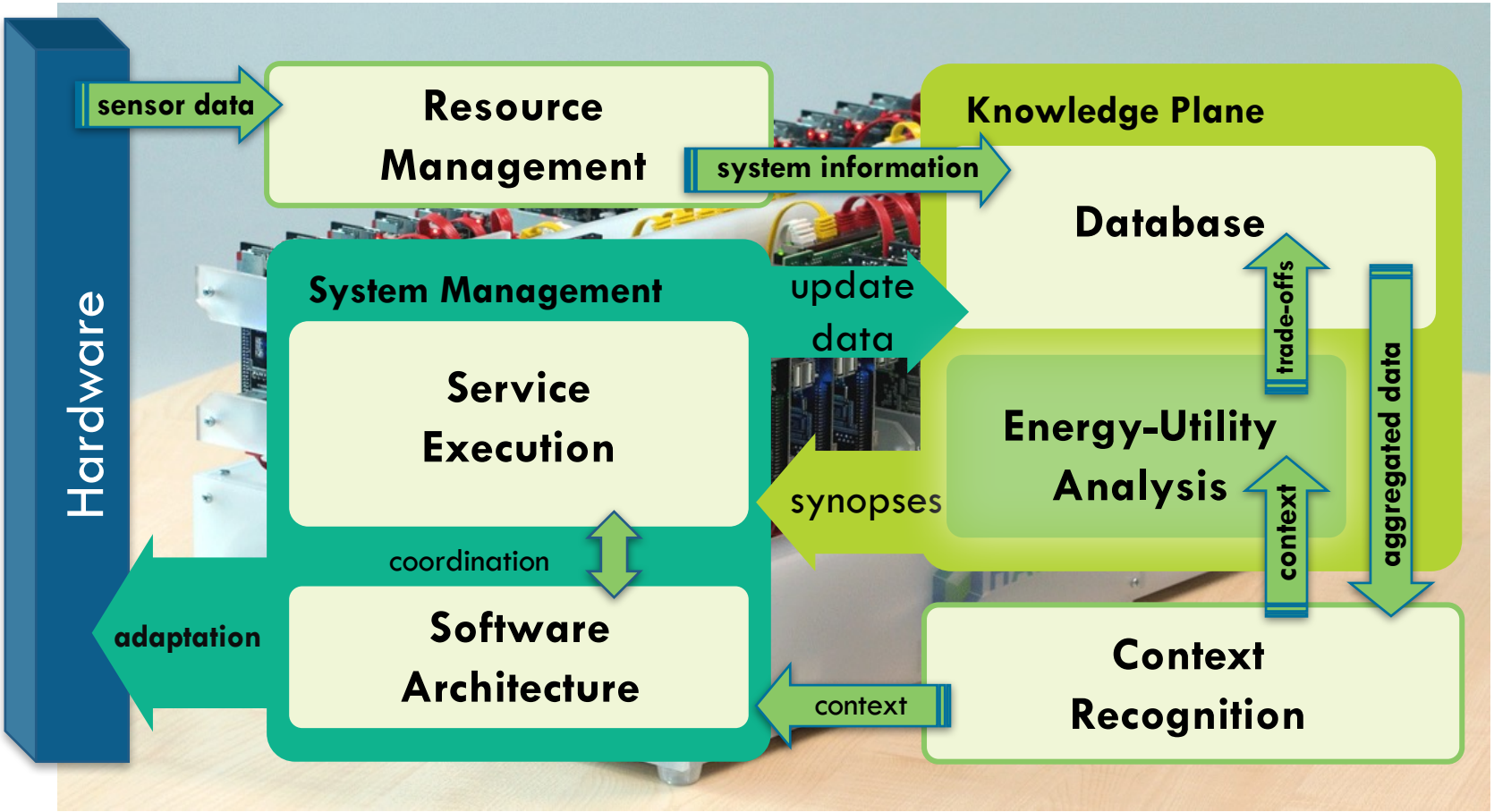
**HAEC-HW**

# HAEC: HW

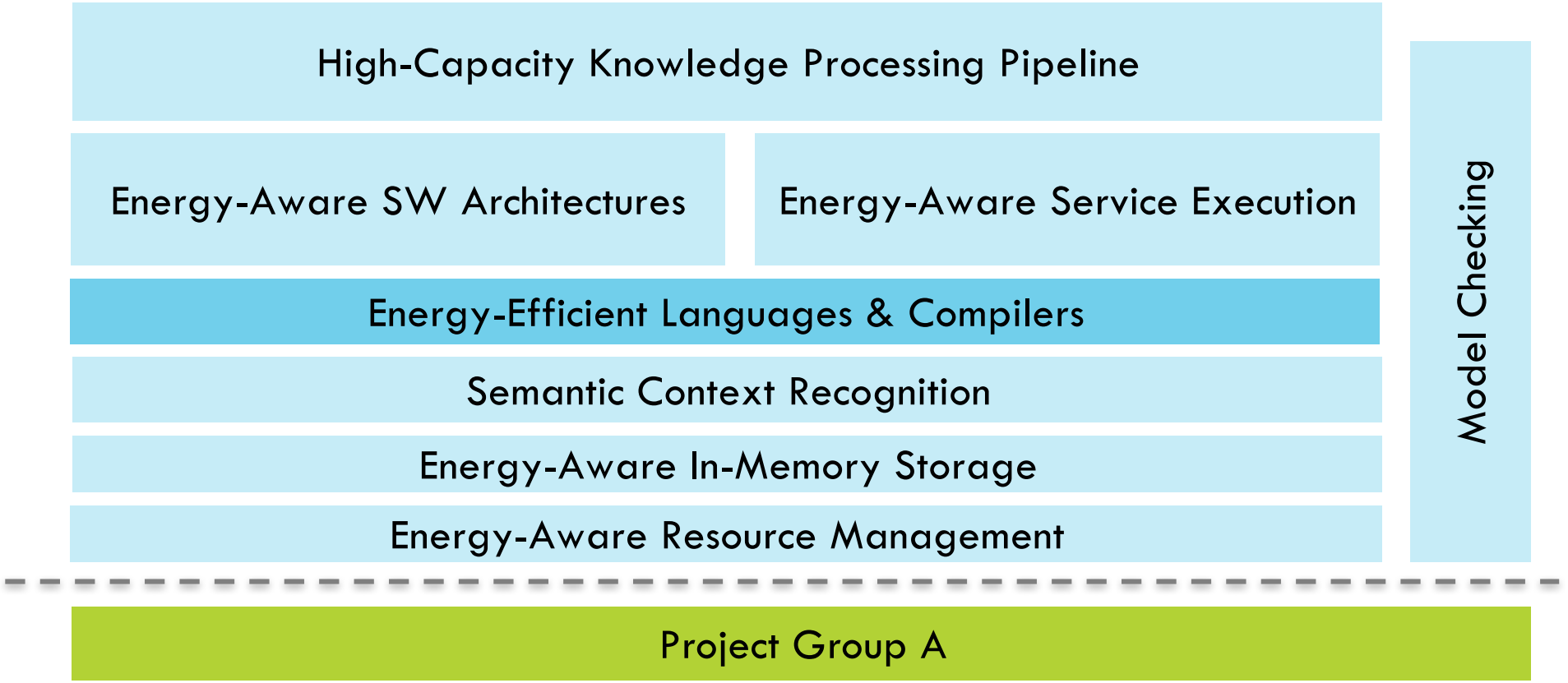


# HAEC SW – Overview

# HAEC SW – Overview



# HAEC: SW – Stack

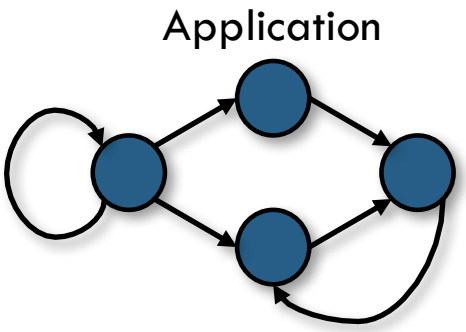


# Agenda

- Contextualization
- HAEC overview
- **Dataflow programming**
- Flexible mappings
- Closing remarks

# Dataflow programming

- Graph representation of applications
  - Implicit repetitive execution of tasks
  - Good model for streaming applications
  - Good match for signal processing & multi-media applications



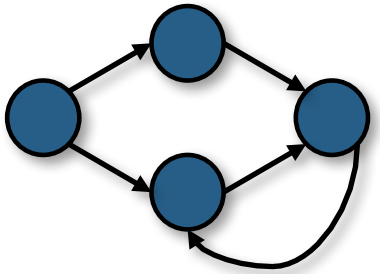

- Flavors expose different trade-offs analyzability-expressiveness
  - Synchronous dataflow (SDF): Static schedules
  - Kahn process networks (KPN): Deterministic execution
  - Dynamic dataflow (DDF): Allows non-determinism and peeking into channels

# Dataflow programming: Examples

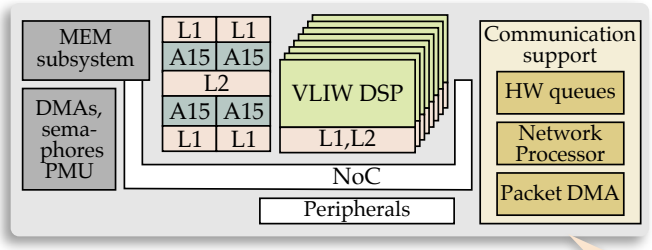
- There are multiple tool flows, languages and frameworks
- Embedded
  - MAPS (now Silexica): KPNs onto heterogeneous multi-cores [Castrill13a]
  - TURNUS: From DDFs (in CaI) onto heterogeneous, reconfigurable systems [Brunet15]
  - Ptolemy: Analysis of interactions between different models [Ptolemaeus14]
  - Preesm: Parametrized SDFs for signal processing [Pelcat14]
- HPC
  - Maxeler: Dataflow cores on FPGA fabric [Maxeler]
  - OpenMP 4.5 (OmpSs, Teraflux) [Duran11, Giorgi14]

# Dataflow programming flow

KPN Application



Architecture model




Iterative analysis, (meta)heuristics, profiling information

Core and memory assignment, scheduling, buffer sizing, …

Analysis

Synthesis

[Castrill11]

Configurations

Source-to-source compilation

Property models (timing, energy, error, …)

Non-functional specification

Models of architectures used for analysis

```
PNargs_ifft_r.ID = 6U;
   PNargs_ifft_r.PNchannel_freq_coef = filtered_coef_r
   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;
```

# Language: C for process networks

- FIFO Channels

```
typedef struct { int i; double d; } my_struct_t;
__PNchannel my_struct_t S;
__PNchannel int A = {1, 2, 3}; /* Initialization */
__PNchannel short C[2], D[2], F[2], G[2];
```

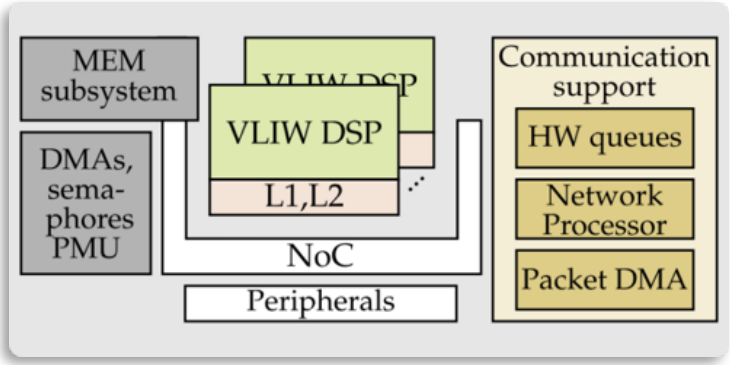
- Processes & networks

```
__PNkpn AudioAmp __PNin(short A[2]) __PNout(short B[2])
                 __PNparam(short boost){
  while (1)
    __PNin(A) __PNout(B) {
      for (int i = 0; i < 2; i++)
        B[i] = A[i]*boost;
    }}
__PNprocess Amp1 = AudioAmp __PNin(C) __PNout(F) __PNparam(3);
__PNprocess Amp2 = AudioAmp __PNin(D) __PNout(G) __PNparam(10);
```

[Sheng14]

# Architecture model for heterogeneity

- System model including:
  - Topology, interconnect, memories
  - Computation: cost tables (as backup)
  - Communication: cost function (no contention)
- Example: Texas Instruments Keystone

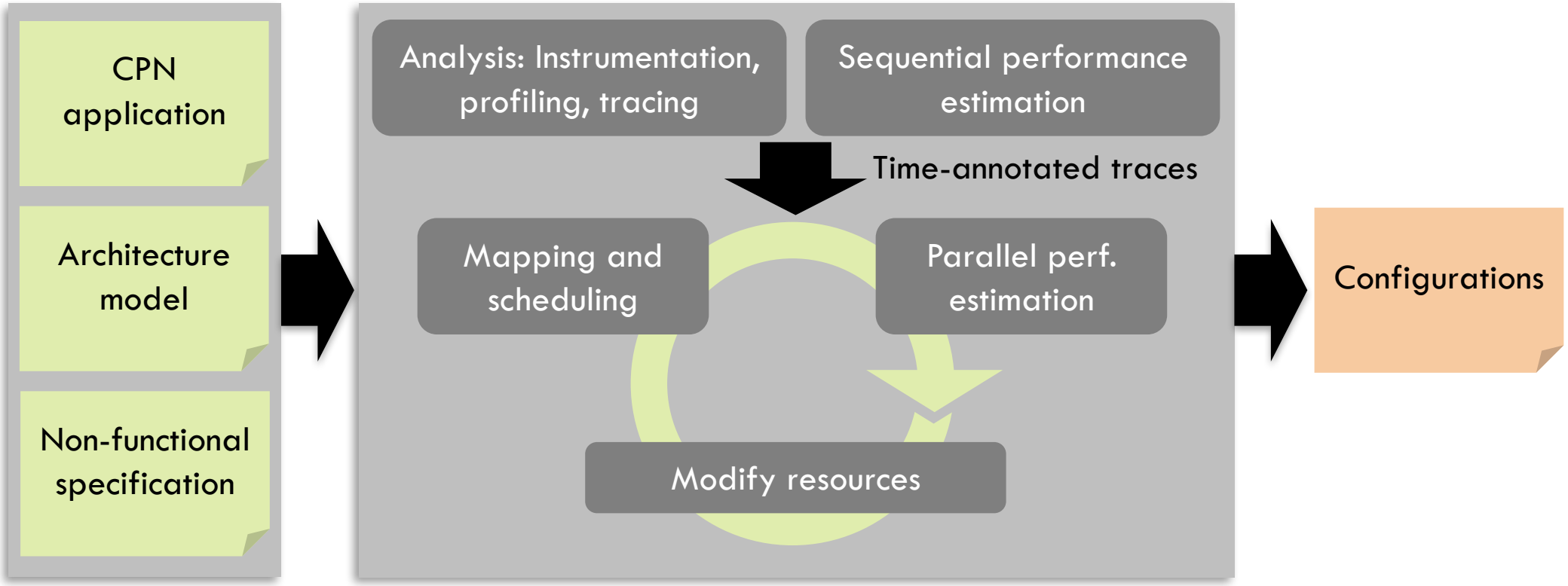



```
-<Platform>
   <Processors List="dsp0 dsp1 dsp2 dsp3 dsp4 dsp5 dsp6 dsp7"/>
   <Memories List="local_mem_dsp0_L2 local_mem_dsp1_L2 local_mem_dsp2_L2
   local_smem_dsp1_L2 local_smem_dsp2_L2 local_smem_dsp3_L2 local_smem_dsp
   local_mem_dsp3_DDR local_mem_dsp4_DDR local_mem_dsp5_DDR local_mem_ds
   <CommPrimitives List="IPCll_SL2 IPCll_DDR EDMA3_SL2 EDMA3_DDR EDMA
 </Platform>
 <Processor Name="dsp0" CoreRef="DSPC66"/>
 <Processor Name="dsp1" CoreRef="DSPC66"/>
    ...
 <Processor Name="dsp7" CoreRef="DSPC66"/>
-<Memory>
   <LocalMemory Name="local_mem_dsp0_L2" Size="524288" BaseAddress_hex="00800000" ProcessorRef="dsp0"/>
 </Memory>
    ...
```

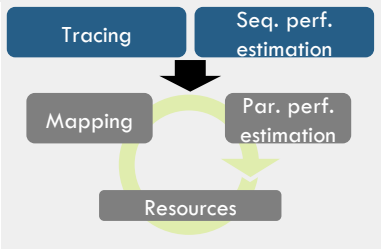
```
-<Core Name="DSPC66" CoreType="DSPC66" Category="DSP">
 -<MultiTaskingInfo MaxNumberOfTasks="-1">
   <ContextSwitchInfo StoreTime="1000" LoadTime="1000"/>
   <SchedulingPolicies List="FIFO PriorityBased"/>
  </MultiTaskingInfo>
 -<CostTable>
  -<Operation Name="Load">
   -<VariableType Name="Char">
     <Cost>1</Cost>
    </VariableType>
   -<VariableType Name="Double">
```

[Oden13]

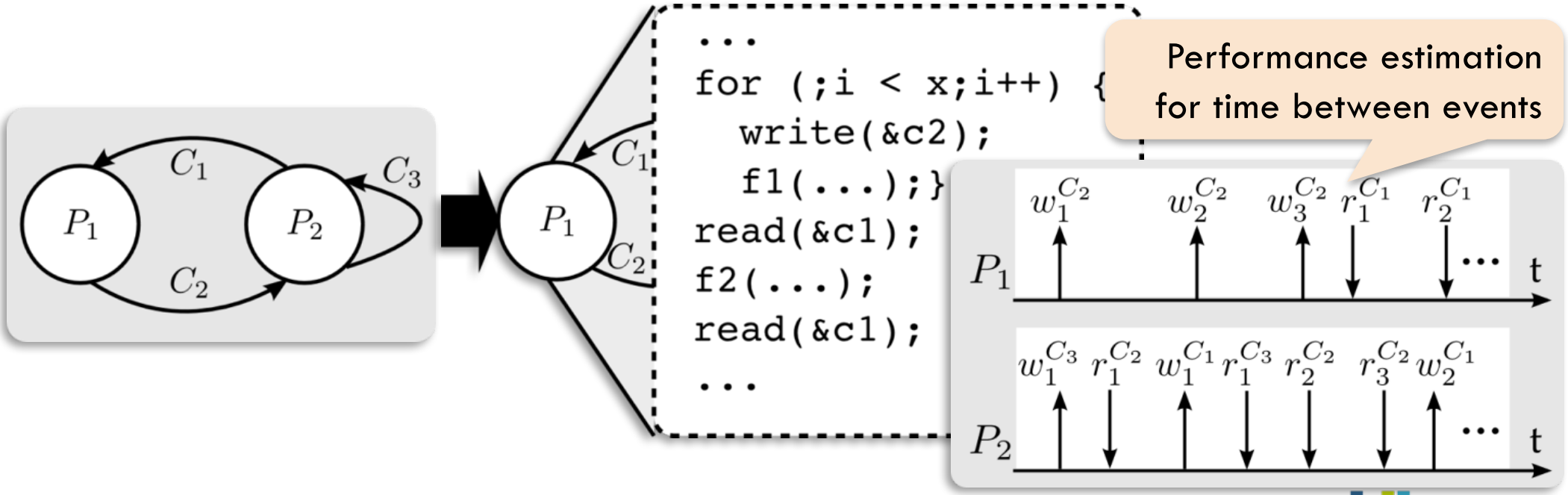
# Analysis and synthesis: Overview



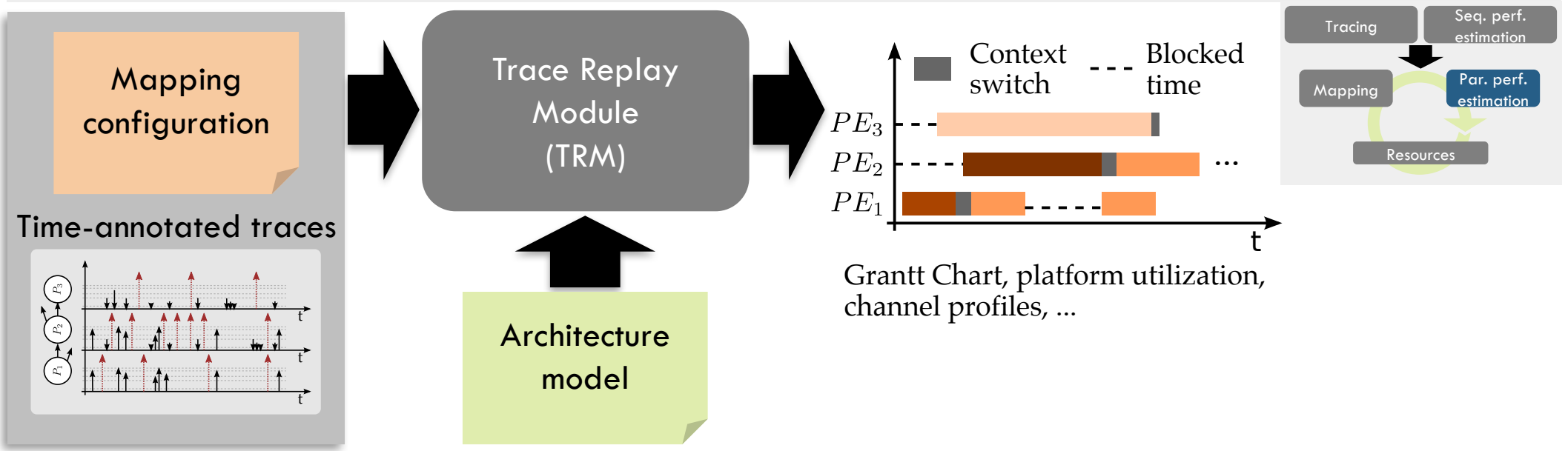
# Tracing: Dealing with dynamic behavior




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

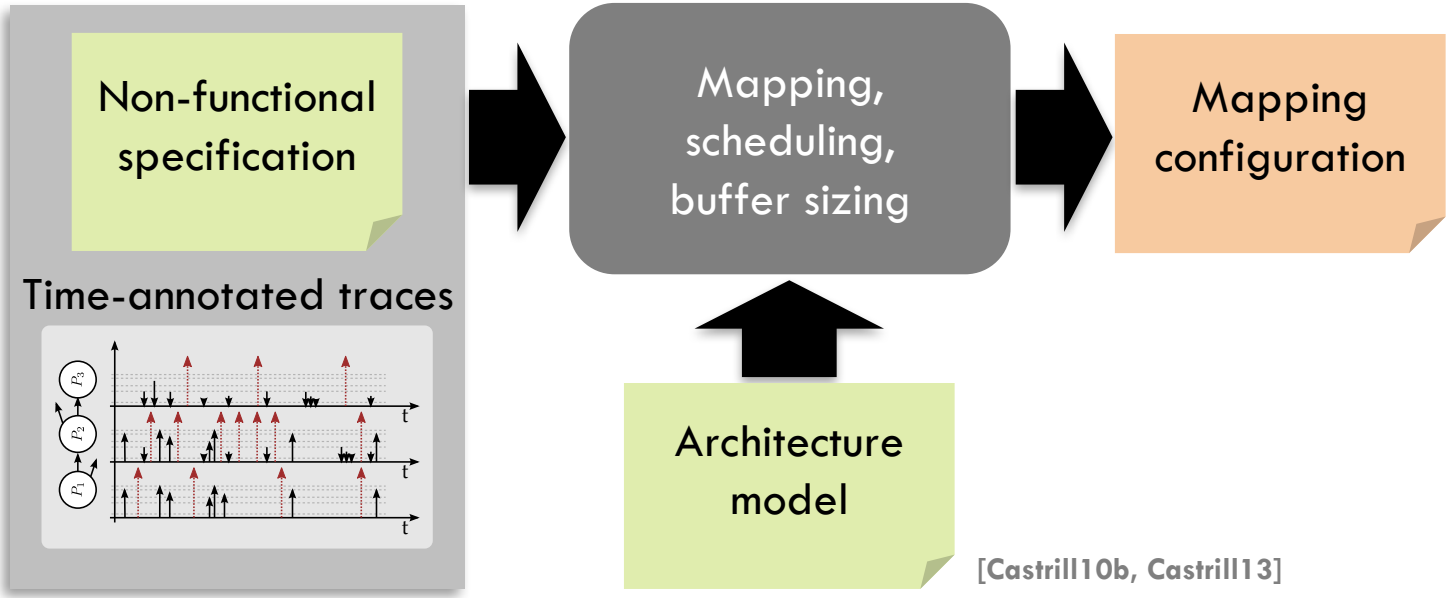


# Parallel performance estimation

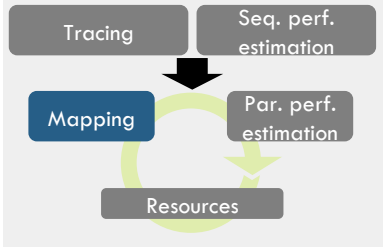



- Discrete event simulator to evaluate a solution
  - Replay traces according to mapping
  - Extract costs from architecture file (NoC modeling, context switches, communication)

# Trace-based synthesis



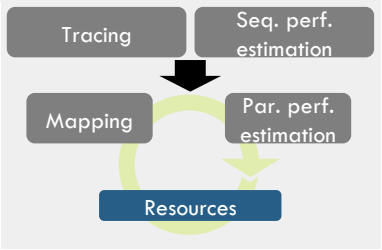

[Castrill10b, Castrill13]

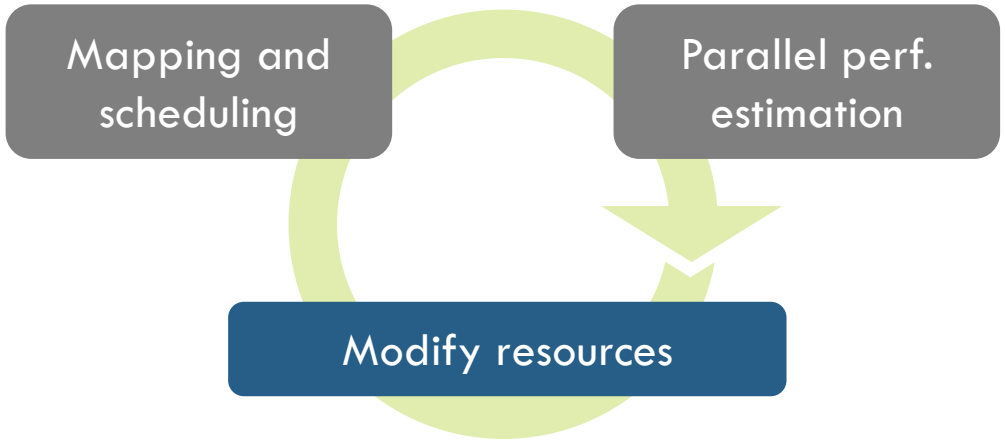



- Multiple heuristics available in the literature
  - Simple/fast heuristics based on the traces
  - Evolutionary algorithms

# Generating variants

- Modify resources available to the synthesis to generate variants

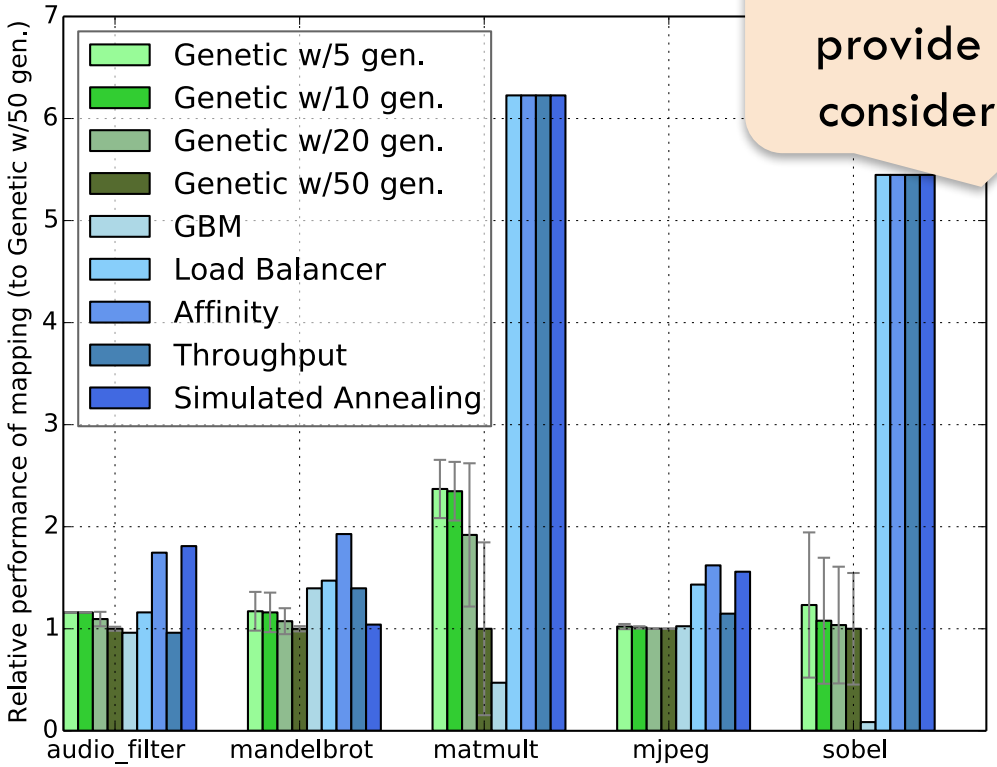


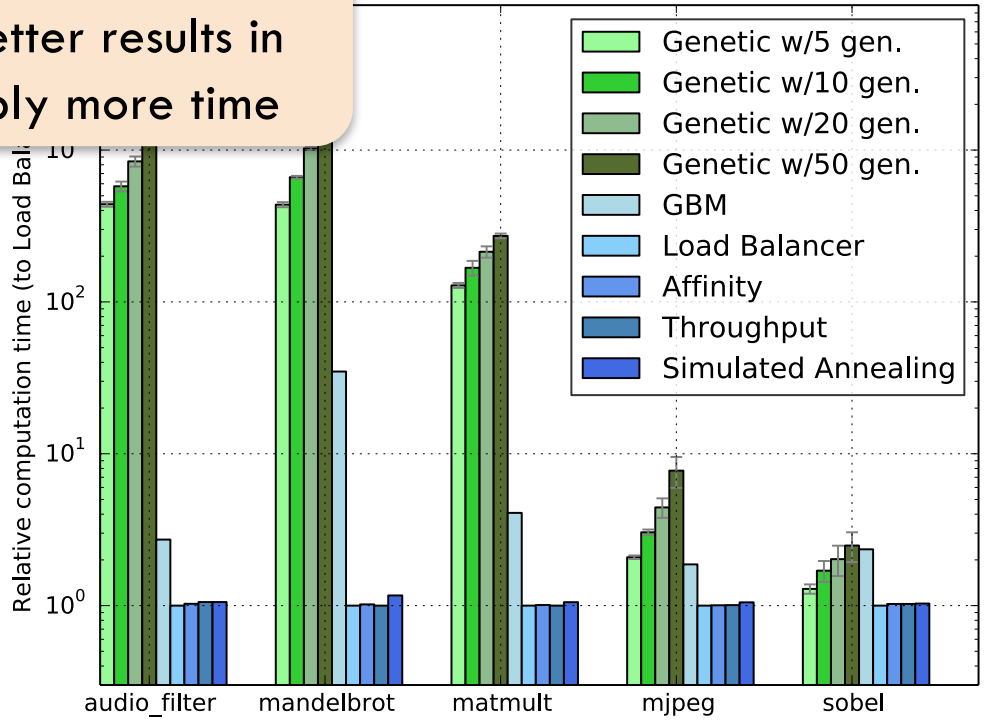




- For homogeneous platforms: add processors
- For heterogeneous: Try out different combination of resources

# Selected results: Heuristics (TI Keystone II)

Genetic algorithms provide better results in considerably more time




[Goens16]

# Selected results: Variant generation

- Sample configurations for JPEG codec on Odroid XU4 (big.LITTLE)

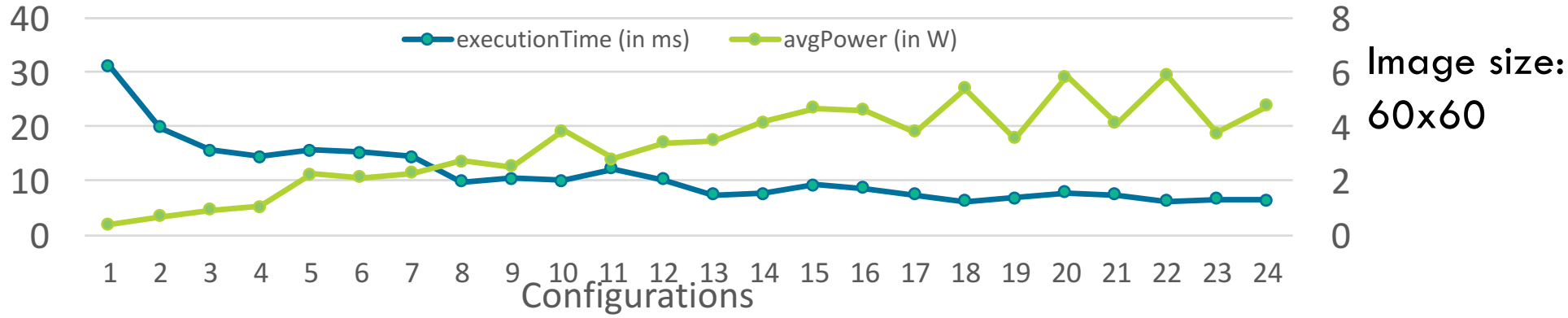



Image size: 60x60

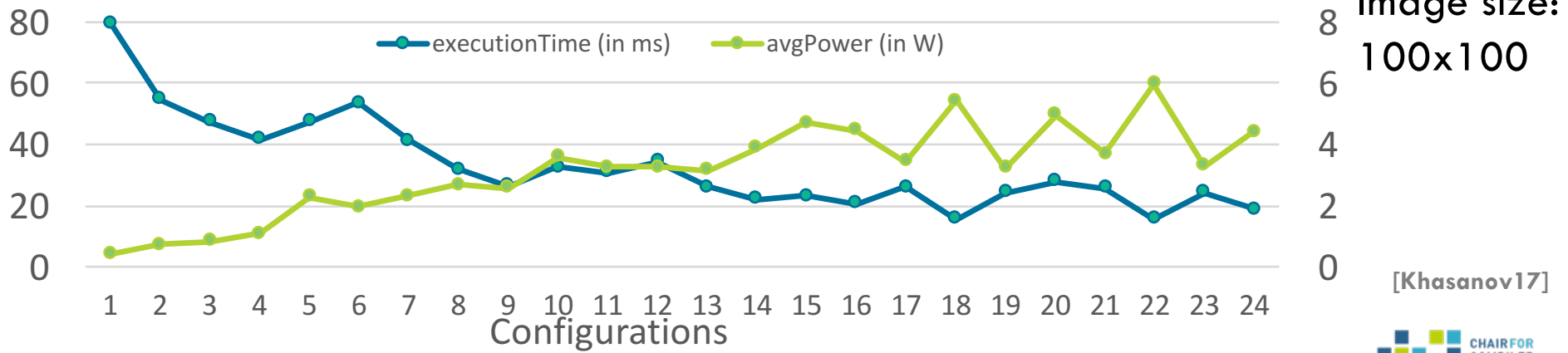


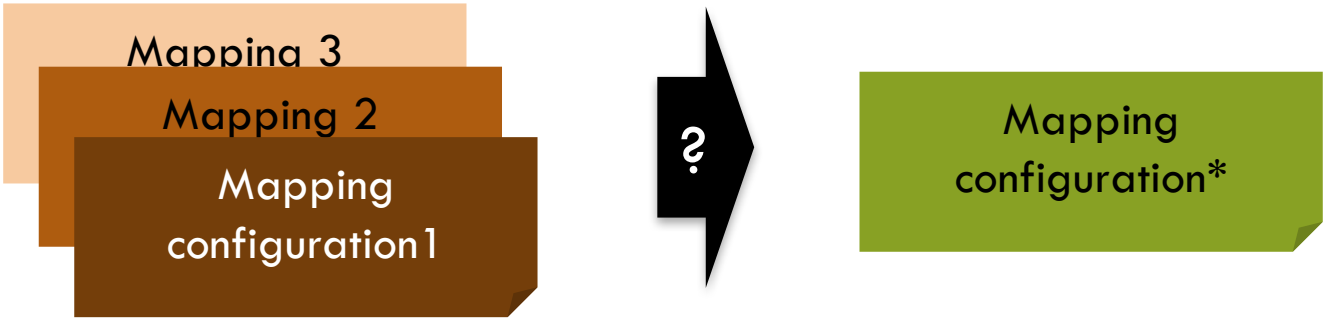

Image size: 100x100

[Khasanov17]

# Agenda

- Contextualization
- HAEC overview
- Dataflow programming
- **Flexible mappings**
- Closing remarks

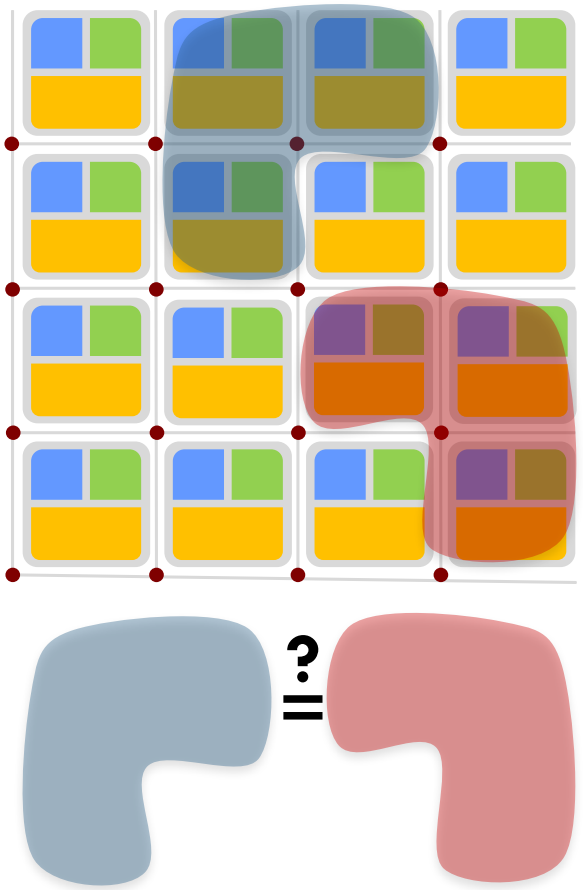
# Flexible mappings




- Given multiple configurations from a compiler, select one at run-time
- Requires: reasoning about mapping **equivalences** and **similarities**

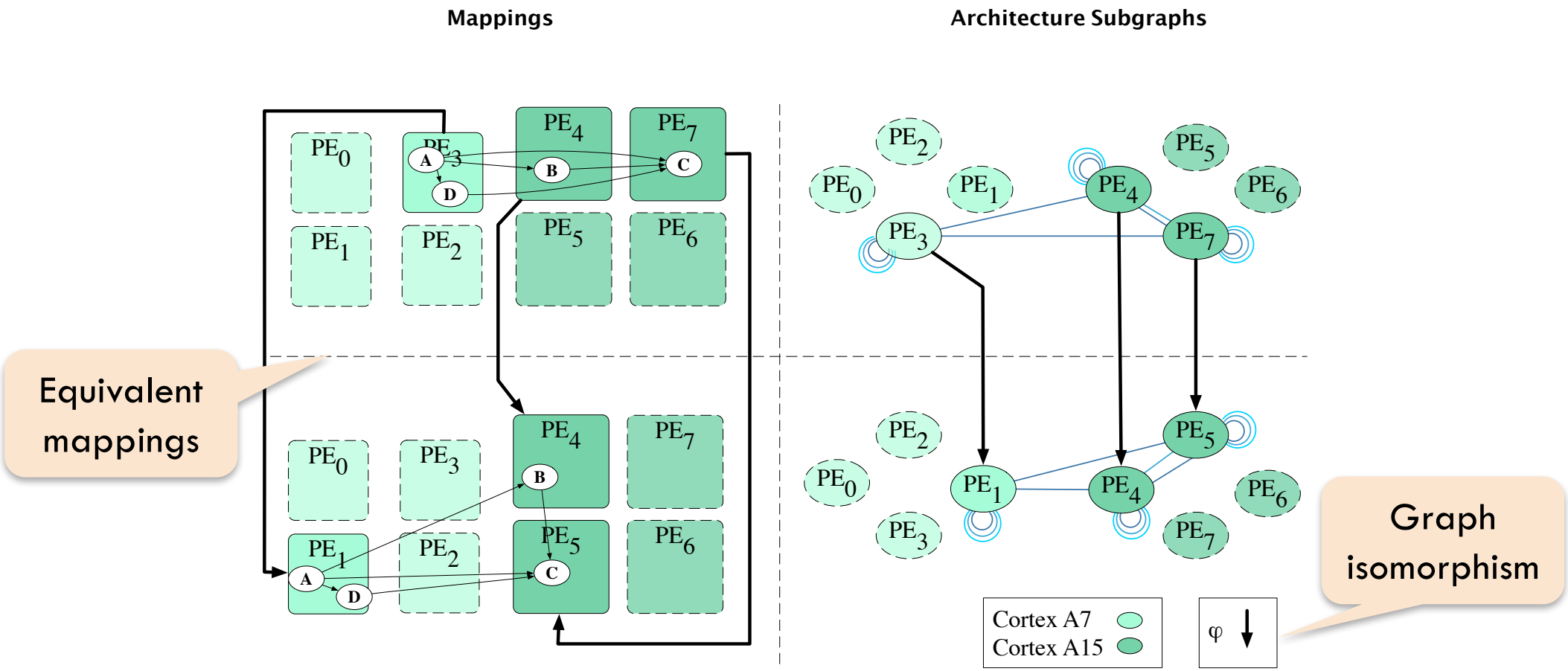
# Mapping equivalences

- Requires analysis of both hardware and software *symmetries*
- Symmetry: Allows "transformation" w/o changing the "outcome"
- Interesting application of group theory (and inverse semi-groups)
- Boils down to the graph isomorphism problem




[Goens15, Goens17a]

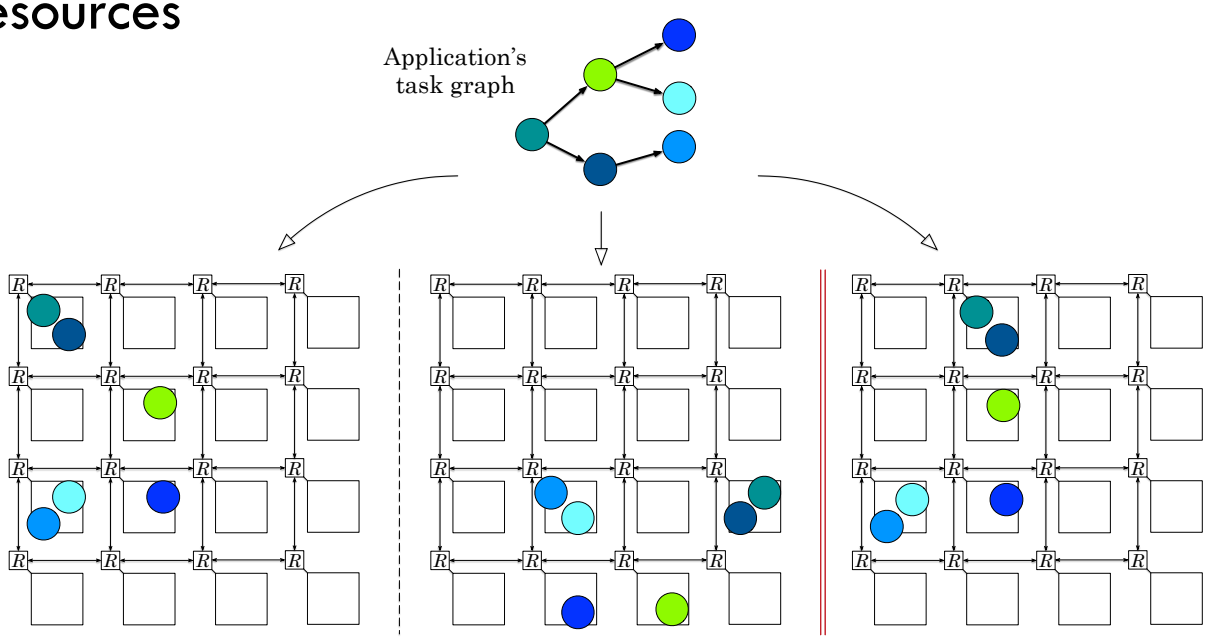
# Symmetries in Odroid: Example




[Goens17b]

# Flexible mappings: Consequences

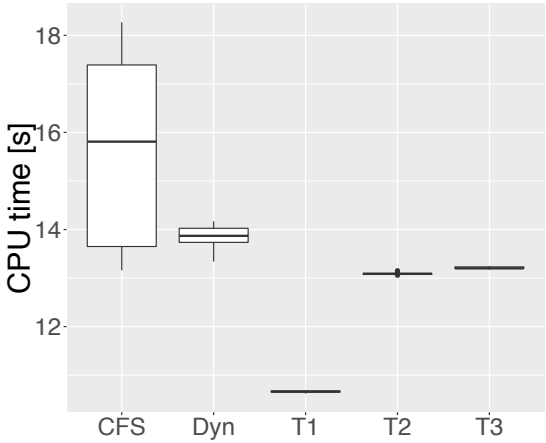
- Scalability: Automatically prune search space (needed for some metaheuristics)
- Run-time adaptivity: Use at runtime to select an equivalent mapping based on available resources



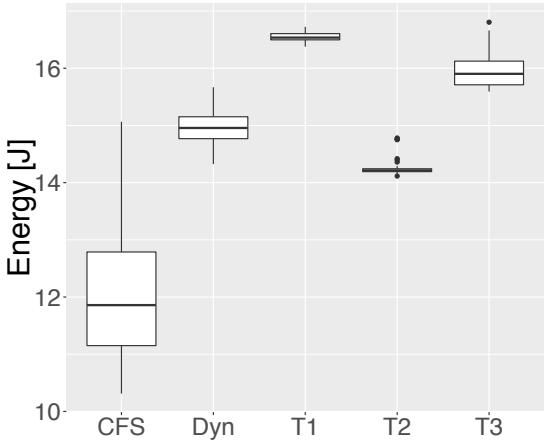

[Goens17b]

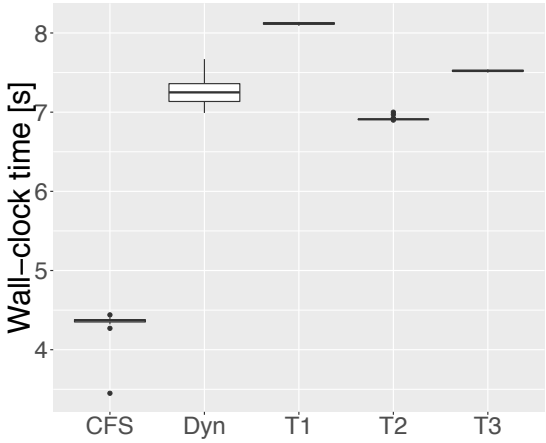
# Flexible mappings: Run-time analysis

- Modified linux kernel: symmetrie-aware
- Target: Odroid XU4 (big.LITTLE)
- Multi-application scenarios: audio filter (AF) and MIMO
  - 1x AF,
  - 4 x AF
  - 2 x AF + 2 x MIMO
- 3 mappings to two processors
  - T1: Best CPU time
  - T2: Best wall-clock time
  - T3: GBM heuristic [Castrill12]



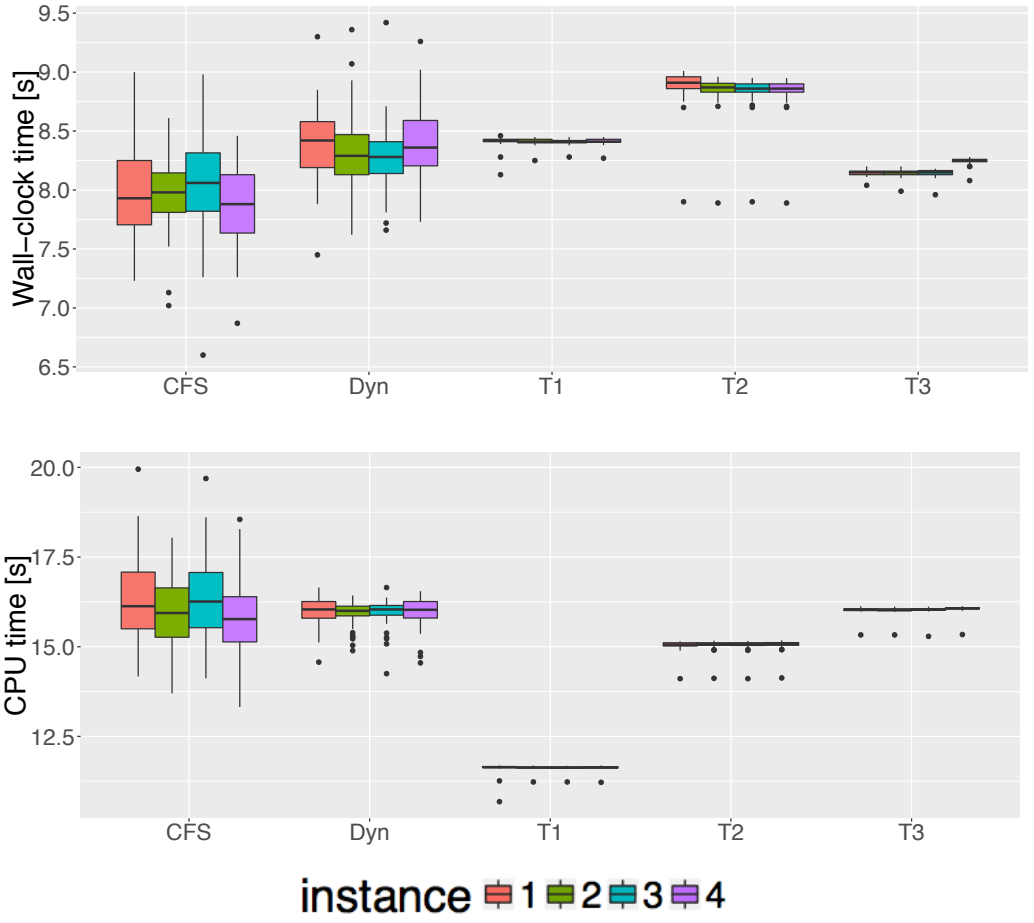

**Single AF**

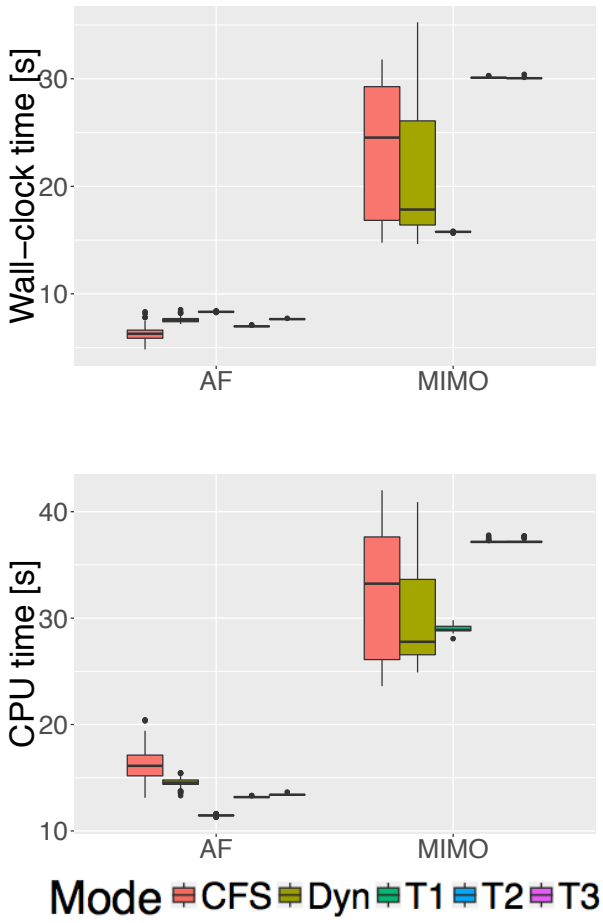



[Goens17b]

# Flexible mappings: Multi-application results (1)



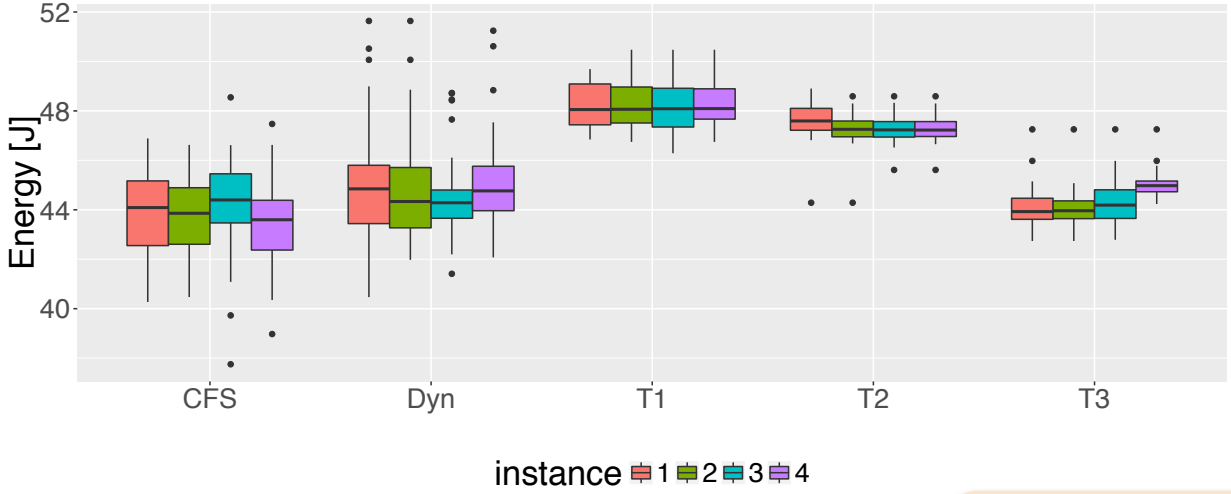



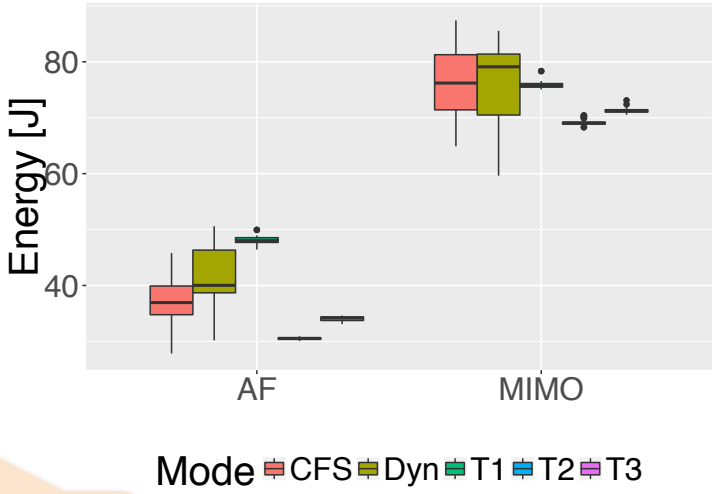

Predictable performance

Comparable performance to dynamic mapping

[Goens17b]

# Flexible mappings: Multi-application results (2)




Good energy predictability as well

[Goens17b]

# Agenda

- Contextualization
- HAEC overview
- Dataflow programming
- Flexible mappings
- **Closing remarks**

# Closing remarks

- Overview of cfaed and HAEC projects
- HAEC
  - Adaptability in HW: Exploit communication paths (e.g., beam-forming, optical)
  - Adaptability in SW: At different levels of the stack / SW development process
- Languages and compilers
  - Dataflow and SW-synthesis methodologies
  - Heuristics to generate multiple variants
  - Run-time strategies to select and transform variants
- Now: Looking into multi-board and flexible communication schemes

# Acknowledgements


- Silexica GmbH
- Collaborative Research Center for Highly-adaptive and Energy-efficient Computing (HAEC)
- German Cluster of Excellence: Center for Advancing Electronics Dresden (www.cfaed.tu-dresden.de)

# References


**[Brunet15]** Brunet, S. C. Analysis and optimization of dynamic dataflow programs. Ecole Polytechnique Federale de Lausanne (EPFL), Ecole Polytechnique Federale de Lausanne (EPFL), 2015.

**[Castrill10]** J. Castrillon, et al. "Trace-based KPN composability analysis for mapping simultaneous applications to MPSoC platforms", Proceedings of the Conference on Design, Automation and Test in Europe, pp. 753-758, 2010

**[Castrill13]** J. Castrillon, R. Leupers, and G. Ascheid, "MAPS: Mapping concurrent dataflow applications to heterogeneous MPSoCs," IEEE Transactions on Industrial Informatics, vol. 9, no. 1, pp. 527–545, 2013

**[Castrill13a]** J. Castrillon, "Programming Heterogeneous MPSoCs: Tool Flows to Close the Software Productivity Gap". Dissertation RWTH Aachen University, 2013.

**[Castrill12]** J. Castrillon, et al. "Communication-aware mapping of KPN applications onto heterogeneous MPSoCs," in Proceedings of the Design Automation Conference, 2012

**[Castrill11]** Castrillon, J.; Sheng, W. & Leupers, R. "Trends in Embedded Software Synthesis", Proceedings of the International Conference on Embedded Computer Systems (SAMOS), 2011, 347-354

**[Duran11]** A. Duran, et al. "Ompss: a proposal for programming heterogeneous multi-core architectures", Parallel Processing Letters 21, 02 (2011), 173–193.

**[Giorgi14]** R. Giorgi, et al. "TERAFLUX: Harnessing dataflow in next generation teradevices", Microprocessors and Microsystems (2014).

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

**[Goens16]** A. Goens, A. et al. "Why Comparing System-level MPSoC Mapping Approaches is Difficult: a Case Study," Proceedings of the Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC-16), 2016, 281-288

**[Goens17a]** Goens, A. et al. "Symmetry in Software Synthesis". In: ACM Transactions on Architecture and Code Optimization (TACO) (2017)

**[Goens17b]** Goens, A. et al. "TETRiS: a Multi-Application Run-Time System for Predictable Execution of Static Mappings", SCOPES'17

**[Haehnel17]** Haehnel, M., et al. "Application Interference Analysis: Towards Energy-efficient Workload Management on Heterogeneous Micro-Server Architectures," In Proceedings of the Workshop on Big Data in Cloud Performance (DCPerf'17), 2017

**[Maxeler]** http://maxeler.com. Visited Jan, 2016.

**[Oden13]** M. Odendahl, J. Castrillon, V. Volevach, R. Leupers and G. Ascheid, "Split-cost communication model for improved MPSoC application mapping", In International Symposium on System on Chip (SoC) pp. 1-8, 2013

**[Pelcat14]** Pelcat, Maxime, et al. "PREESM: A Dataflow-Based Rapid Prototyping Framework for Simplifying Multicore DSP Programming". EDERC 2014, Milan, Italy.

**[Ptolemaeus14]** Ptolemaeus, C. (Ed.) System Design, Modeling, and Simulation using Ptolemy II Ptolemy.org, 2014

**[Sheng14]** W. Sheng, S. Schürmans, M. Odendahl, M. Bertsch, V. Volevach, R. Leupers, and G. Ascheid, "A compiler infrastructure for embedded heterogeneous MPSoCs", Parallel Comput. 40, 2 (February 2014), 51-68