

# Models of computation for energy-efficient time-aware distributed embedded systems

Jeronimo Castrillon

Chair for Compiler Construction (CCC)

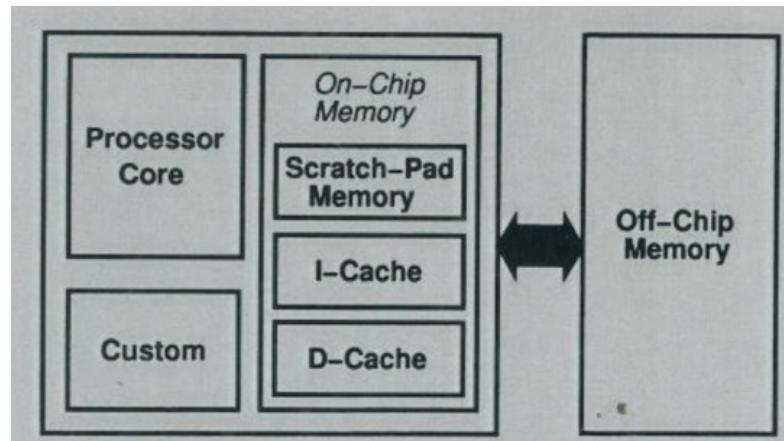
TU Dresden, Germany

International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC'21)  
Singapore University of Technology, Singapore. December 22 2021

# Systems on Chip (SoC): Evolution

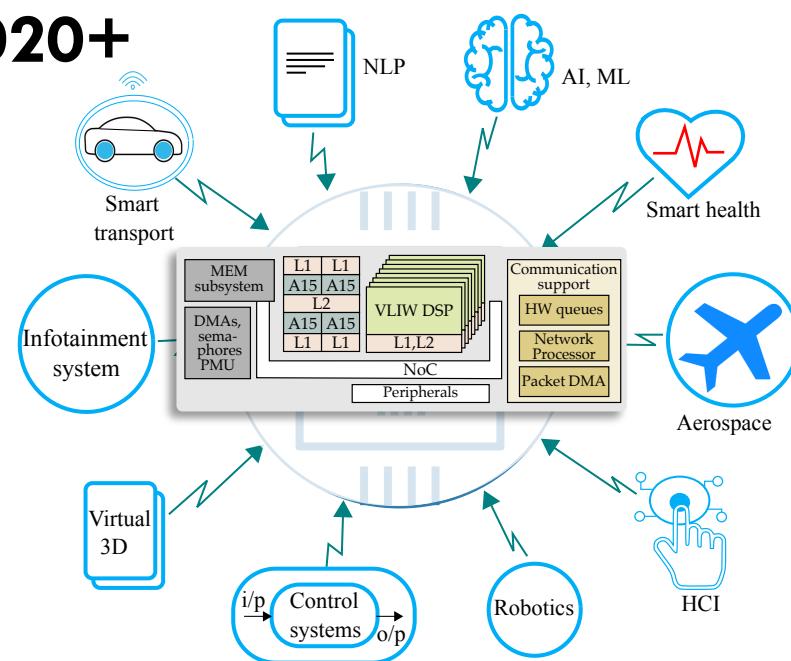
- ❑ Incredible evolution over the last decades
- ❑ SoCs: Long history of specialization and interaction with environment

1999



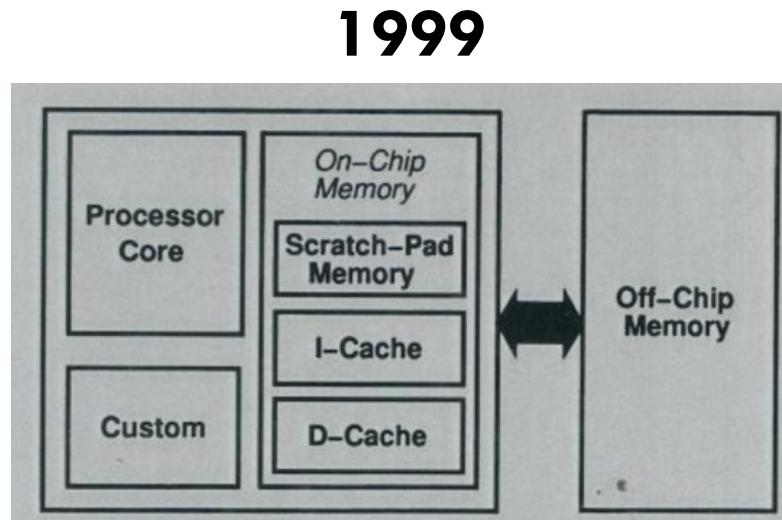
Panda, P. R., Dutt, N. D., & Nicolau, A. Memory issues in embedded systems-on-chip: optimizations and exploration. Springer Science & Business Media. 1999

2020+

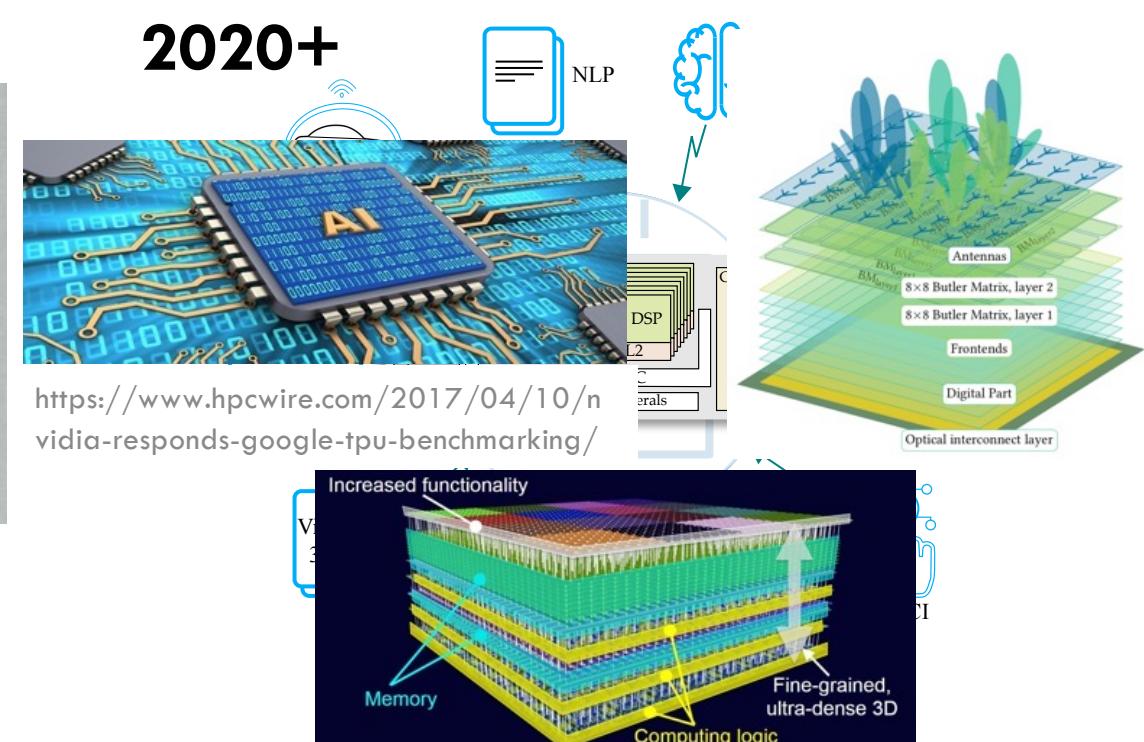


# Systems on Chip (SoC): Evolution

- ❑ Incredible evolution over the last decades
- ❑ SoCs: Long history of specialization and interaction with environment

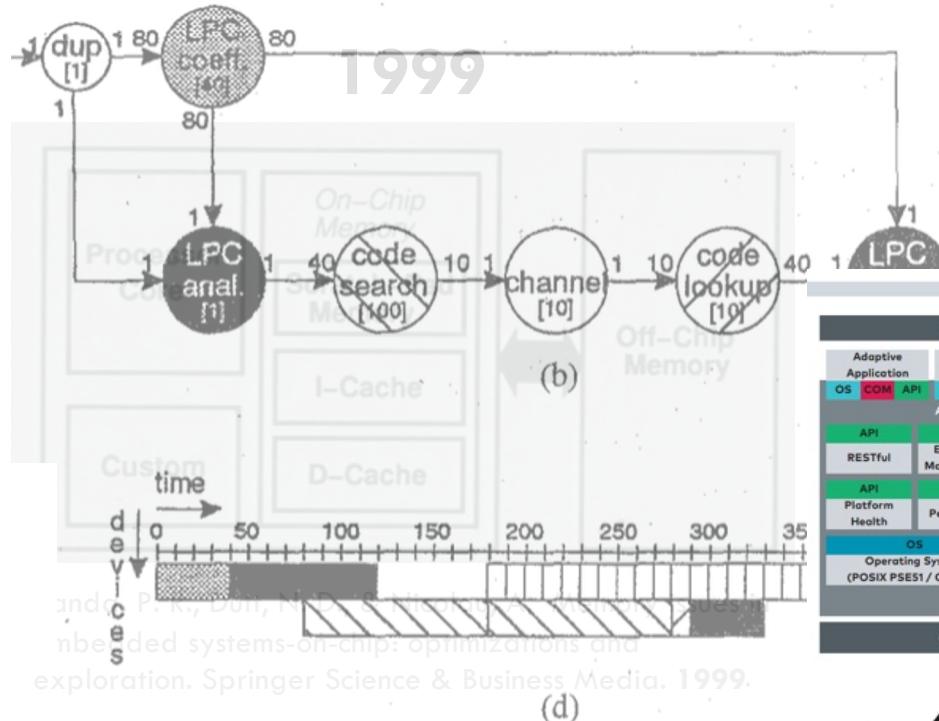


Panda, P. R., Dutt, N. D., & Nicolau, A. Memory issues in embedded systems-on-chip: optimizations and exploration. Springer Science & Business Media. 1999



# Models of computation for 2020+

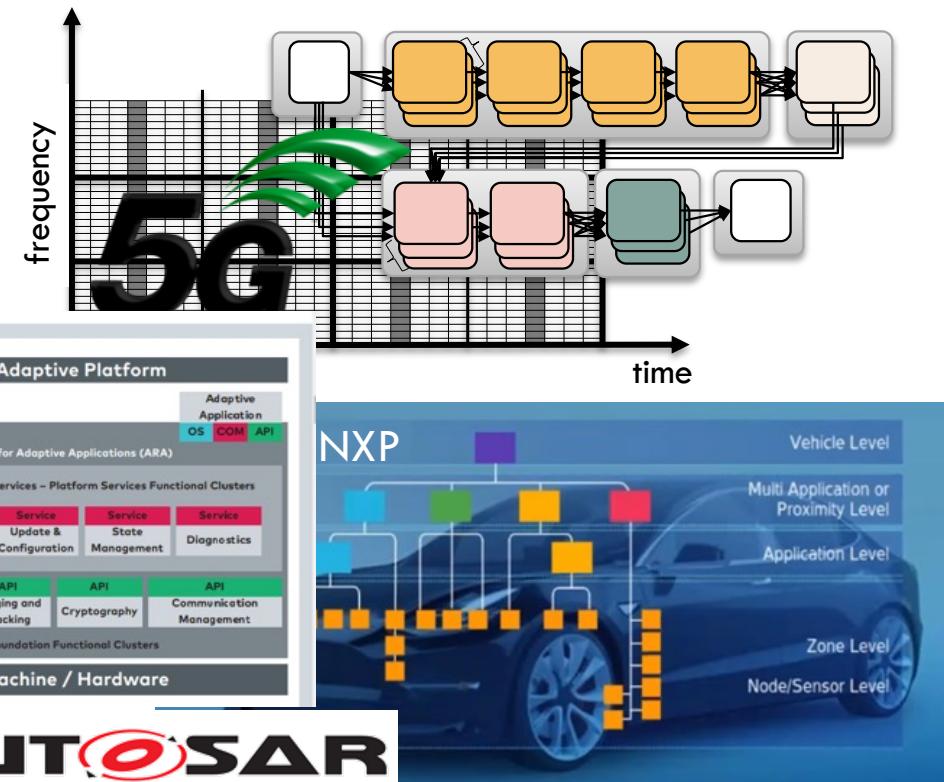
- Need for dynamic graphs, adaptivity, time semantics



(d)

Bilsen, Greet, et al. "Cycle-static dataflow." *Signal Processing, IEEE Transactions on* 44.2 (1996): 397-408.

© Prof. J. Castrillon. MCSoc'21. Singapore, 2021



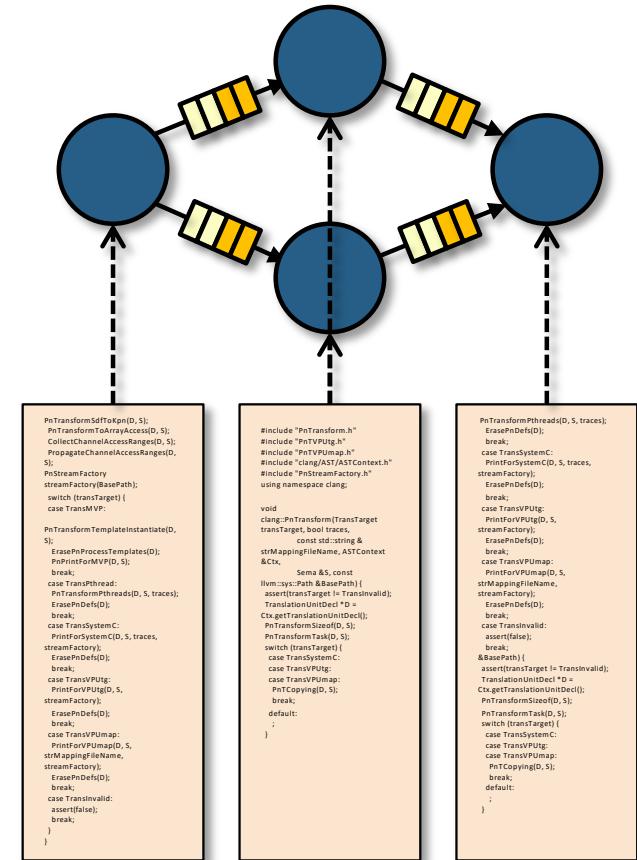
# In this talk



- Introduction
- **Quick background**
- Runtime adaptivity
- CPS it's about time
- Summary

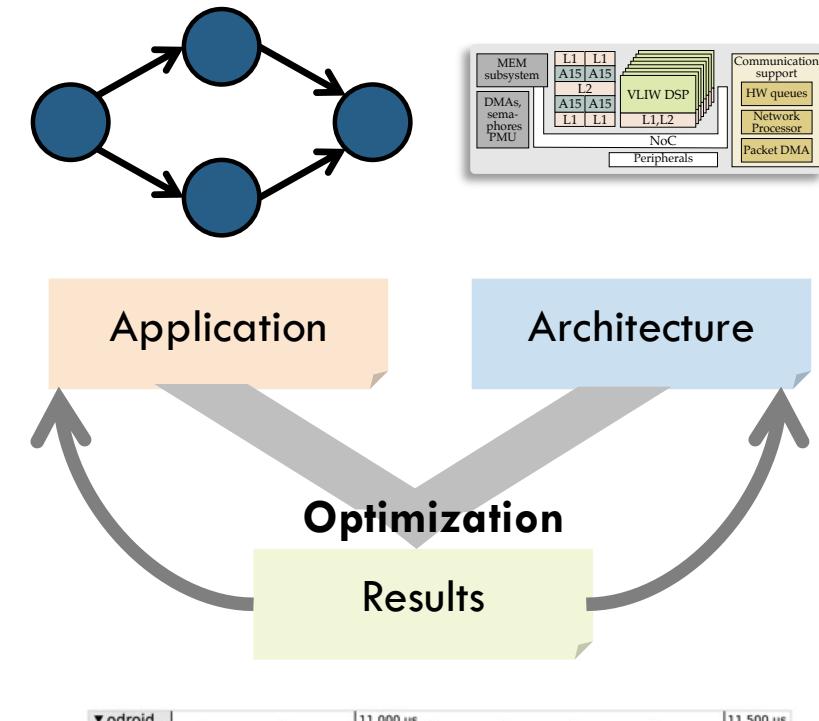
# Dataflow programming

- Graph representation of applications
  - Implicit repetitive execution of tasks
  - Good model for streaming applications
  - Good match for signal processing & multi-media
  
- Some MoCs allow reasoning about
  - Termination, deadlocks
  - Memory consumption
  - Efficient schedules / maximum throughput
  - Determinism



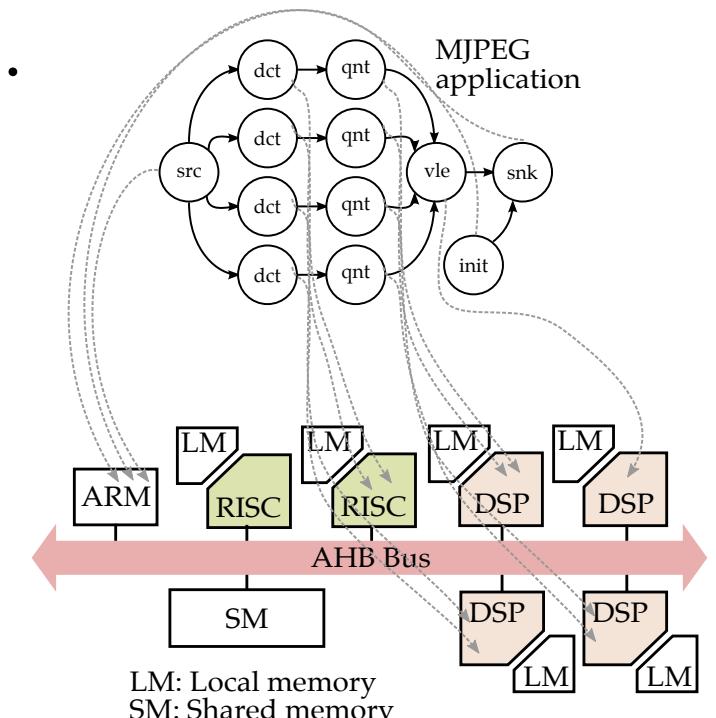
# Classic MoC-based HW-SW co-design

- ❑ Model-based approach
  - ❑ Graph model of application
  - ❑ Graph model of target systems
  - ❑ Early estimation of performance/energy/...
  - ❑ Correct-by-construction code-generation
- ❑ Successful iterative co-design methodologies (since the 80s)
  - ❑ Application
  - ❑ Architecture
  - ❑ Mapping



# Programming flows (static)

- Many: CAL, DOL, CPN, Daedalus, Ptolemy, CAPH, ...
- Information
  - Dataflow model: Rates, states, actions, traces, ...
  - Architecture model: Resources, interconnect, costs, ...
- Optimization: Mapping application to hardware
  - Multi-objective optimization (exact formulations, heuristics, meta-heuristics)
  - High-level simulation / cost models



Castrillon, Springer 2014

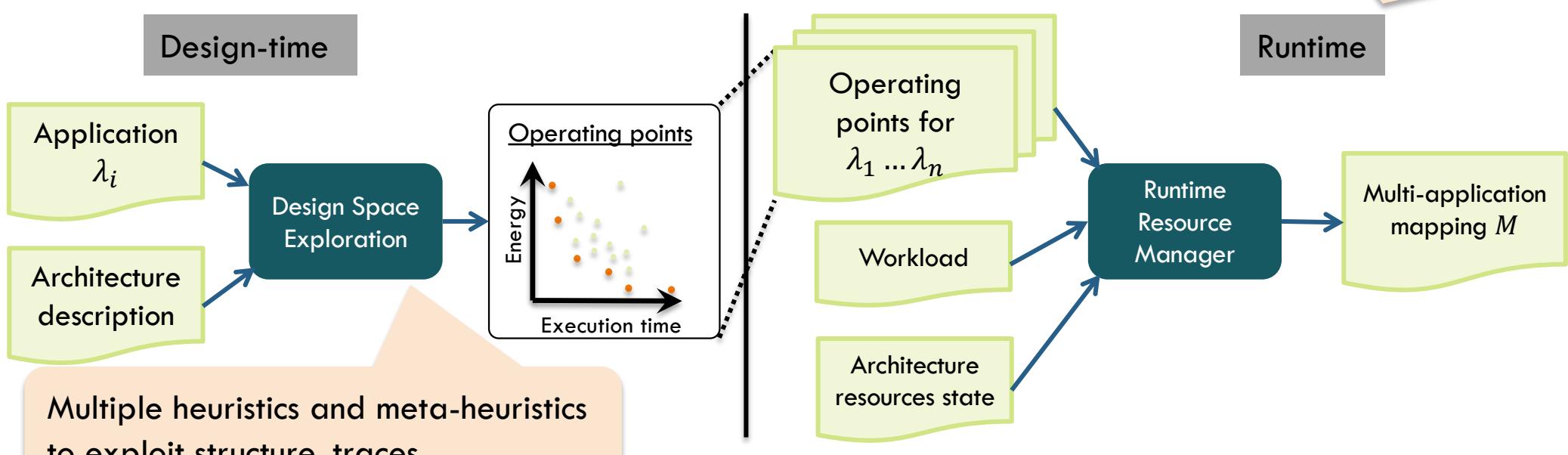
# In this talk



- Introduction
- Quick background
- **Runtime adaptivity**
- CPS it's about time
- Summary

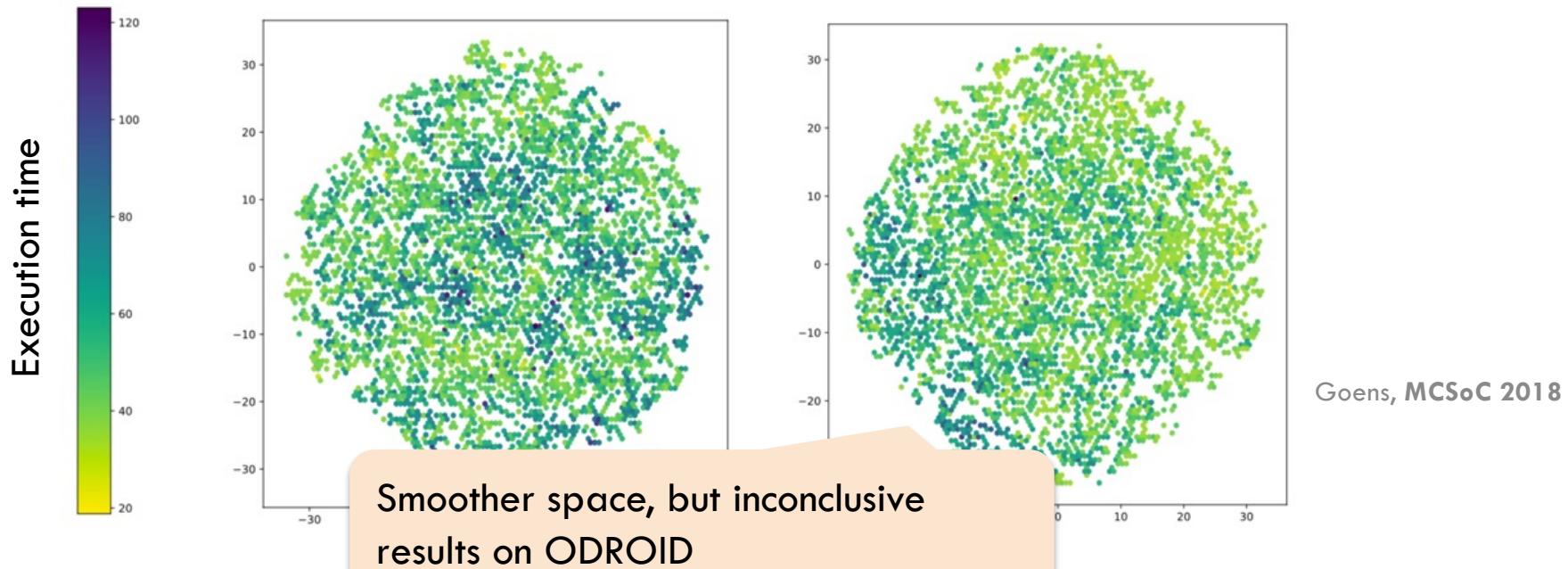
# Hybrid mapping and scheduling

- Hybrid DSE: a compile and run-time approach
  - Enable adaptivity: malleable, multi-variant
  - Run-time predictability, robustness, resource aware



# Coping with complexity: Shape space

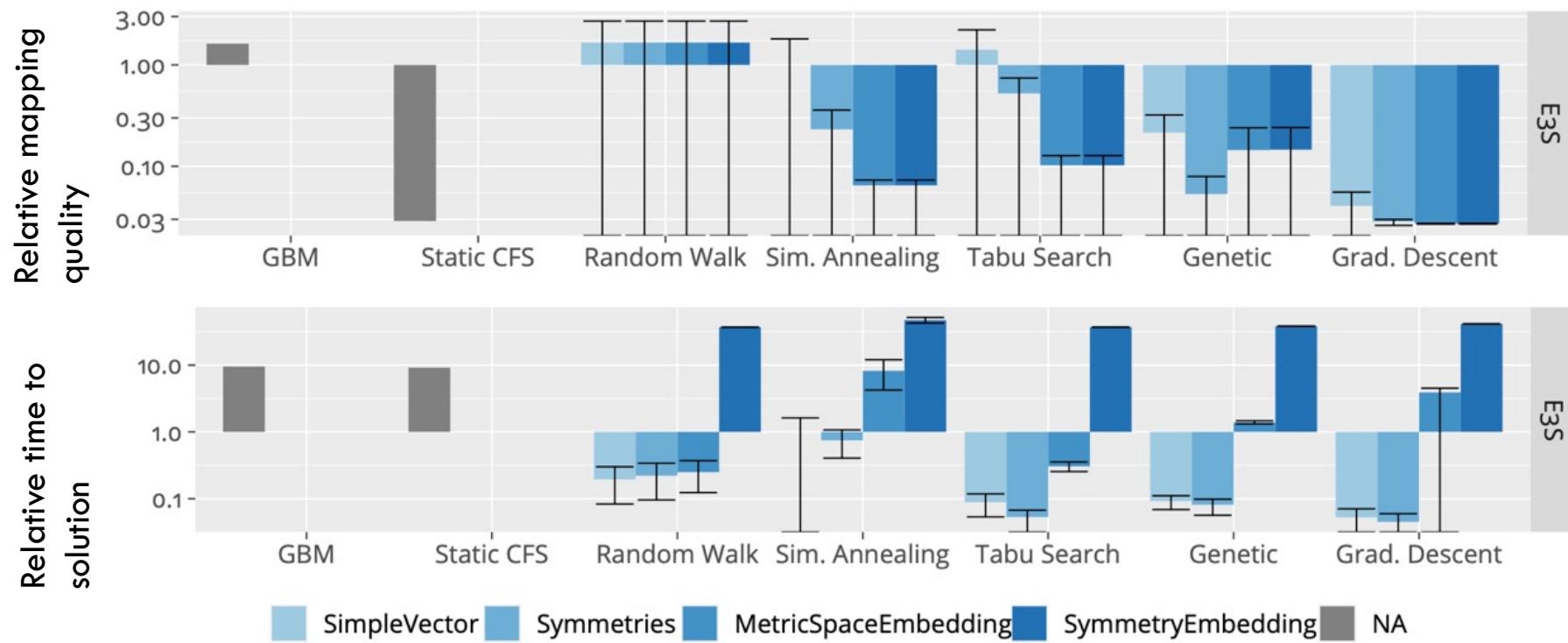
- Embeddings for the mapping space: Improved heuristics?
- Example: T-SNE Visualization for mappings space (8 tasks on Odroid XU4)



## Coping with complexity: Shape space (2)

- Embeddings for the mapping space: Improved heuristics?
- Recent results on the complex MPPA3 Coolidge platform

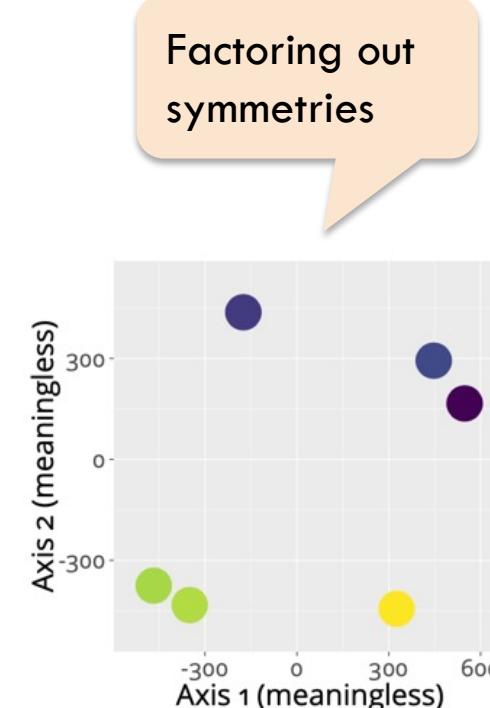
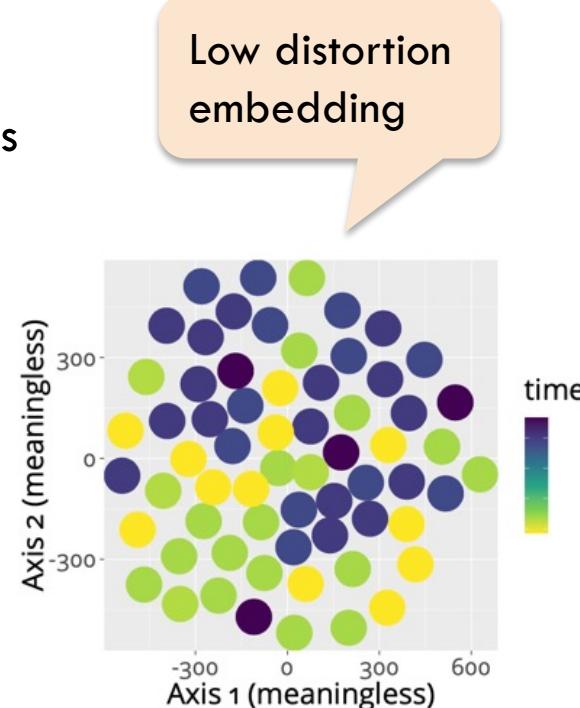
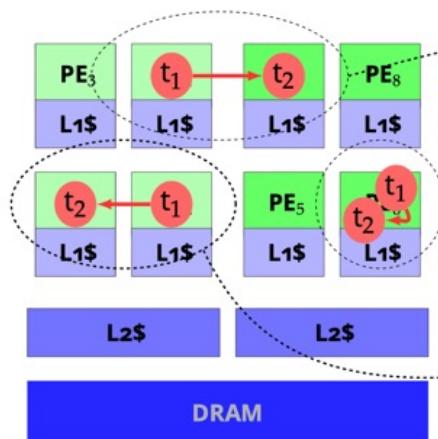
Goens, Castrillon. SAMOS 2021  
Goens, PhD Thesis. 2021



# Coping with complexity: Factoring symmetries out

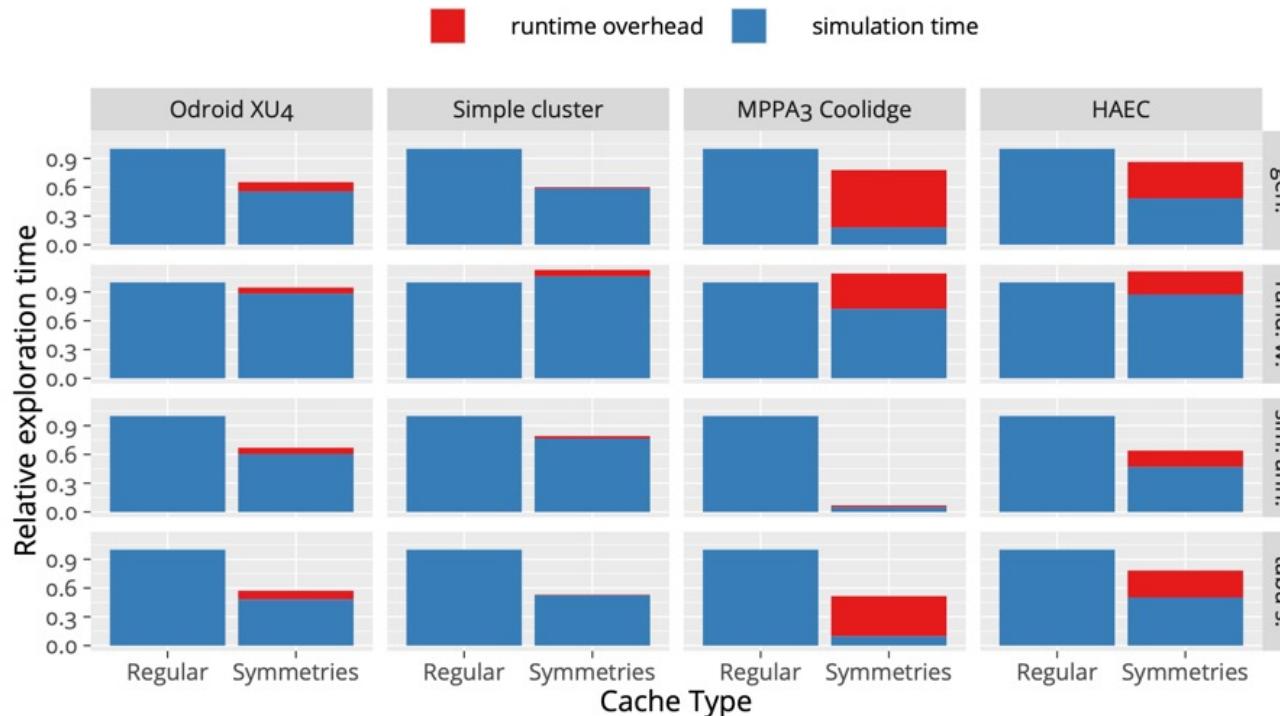
- Symmetries: Performance-invariant mapping transformations
  - Exploit platform symmetries
  - Exploit application symmetries

Goens, PhD Thesis. 2021  
Goens, ACM TACO 2017

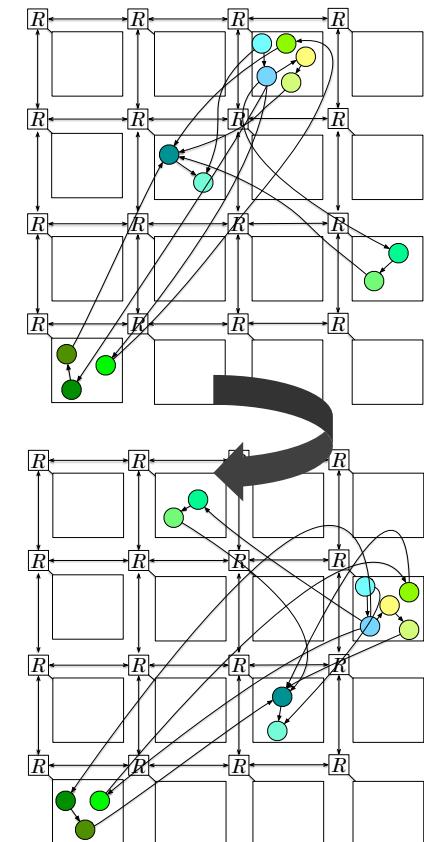


# Coping with complexity: Factoring symmetries (2)

- Efficient algorithms for determining canonical variants, among others

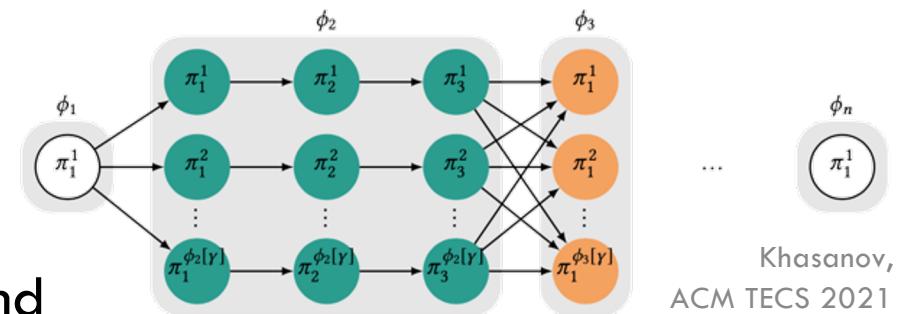


Goens, ACM TACO 2017  
Goens, IEEE TCAD 2021  
Goens, PhD Thesis. 2021

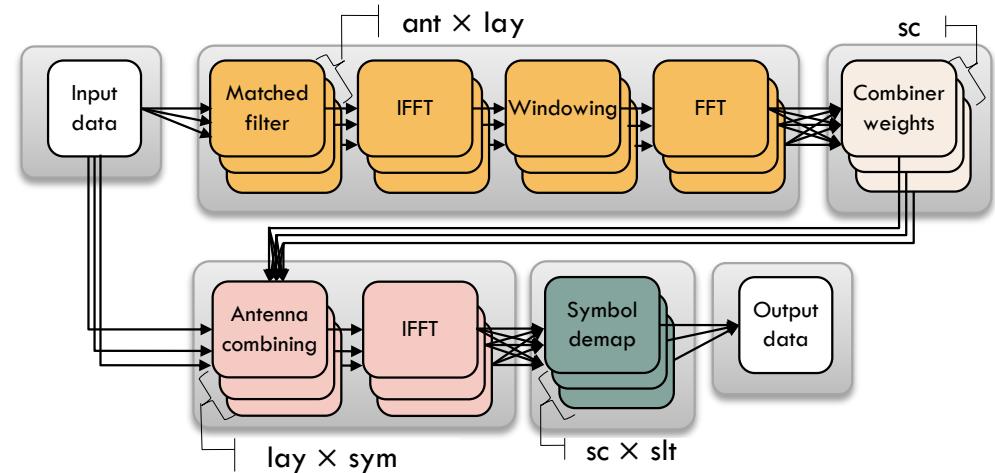
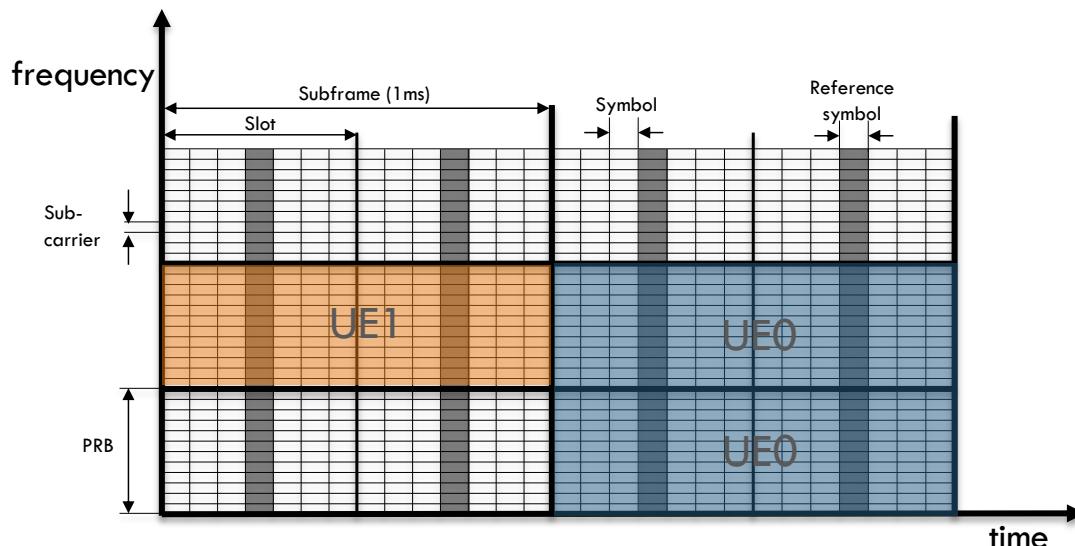


# Coping w/ complexity: Domain-specific knowledge

- Scalability for large graphs: Compose mappings of application phases
- Example: Dynamic graphs in 4/5G baseband

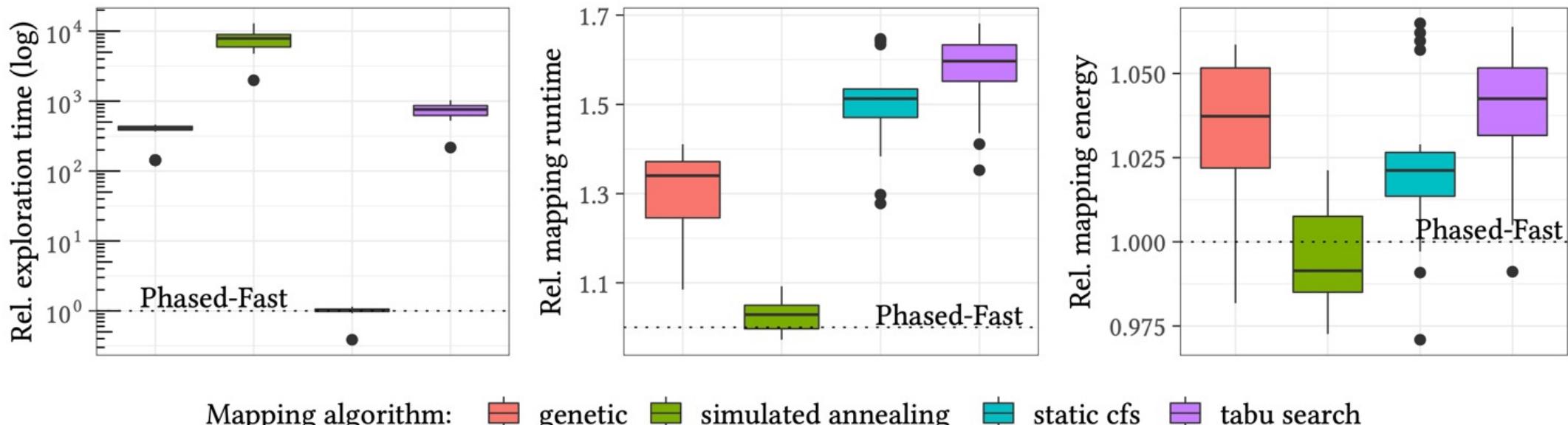


Khasanov,  
ACM TECS 2021



## Coping w/ complexity: Domain-specific knowledge (2)

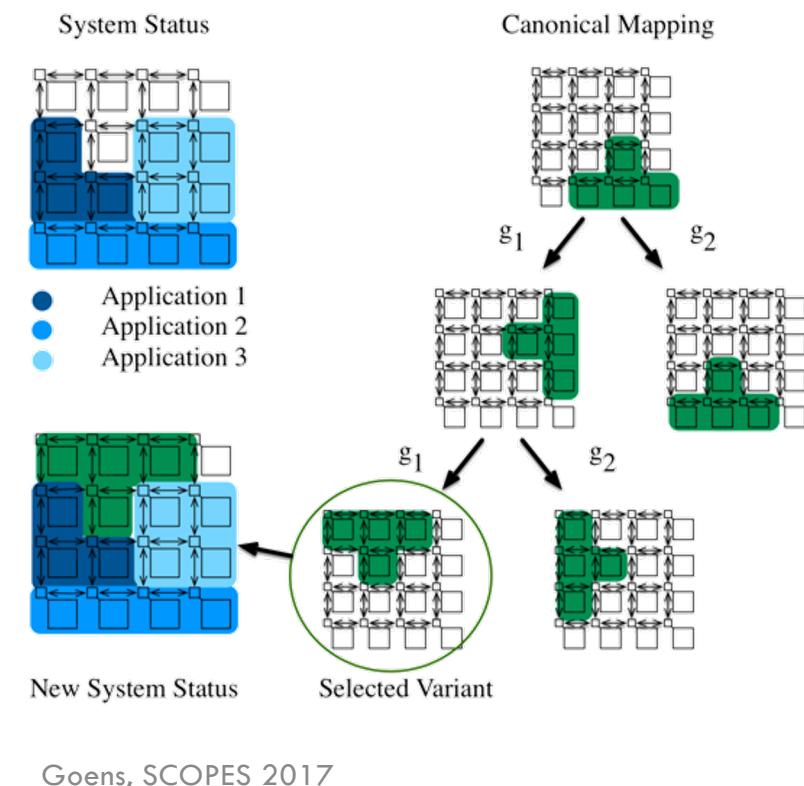
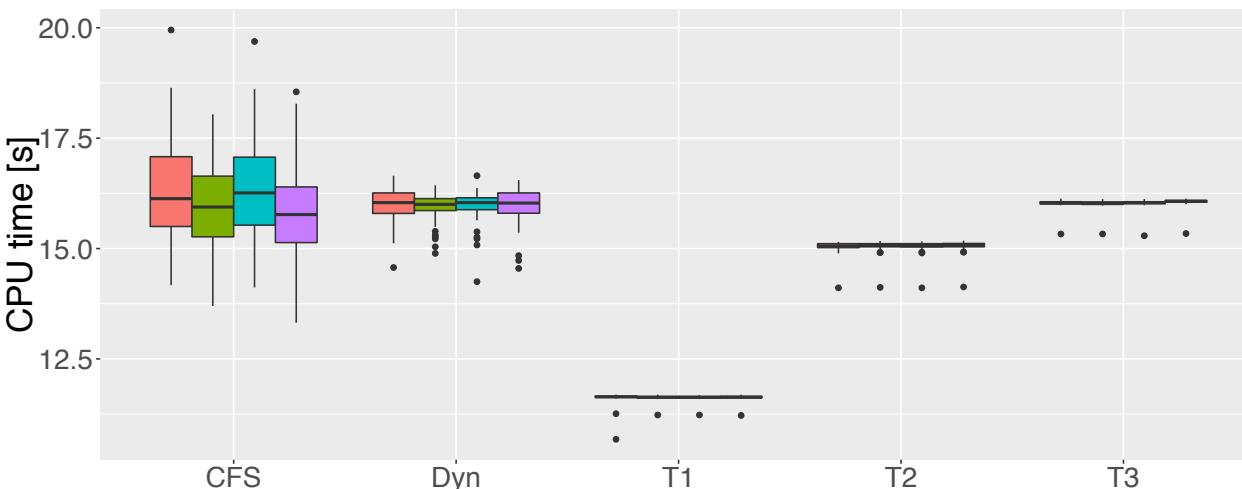
- Exploration time and results for actual 4G traces (graphs) on ODROID



Khasanov,  
ACM TECS 2021

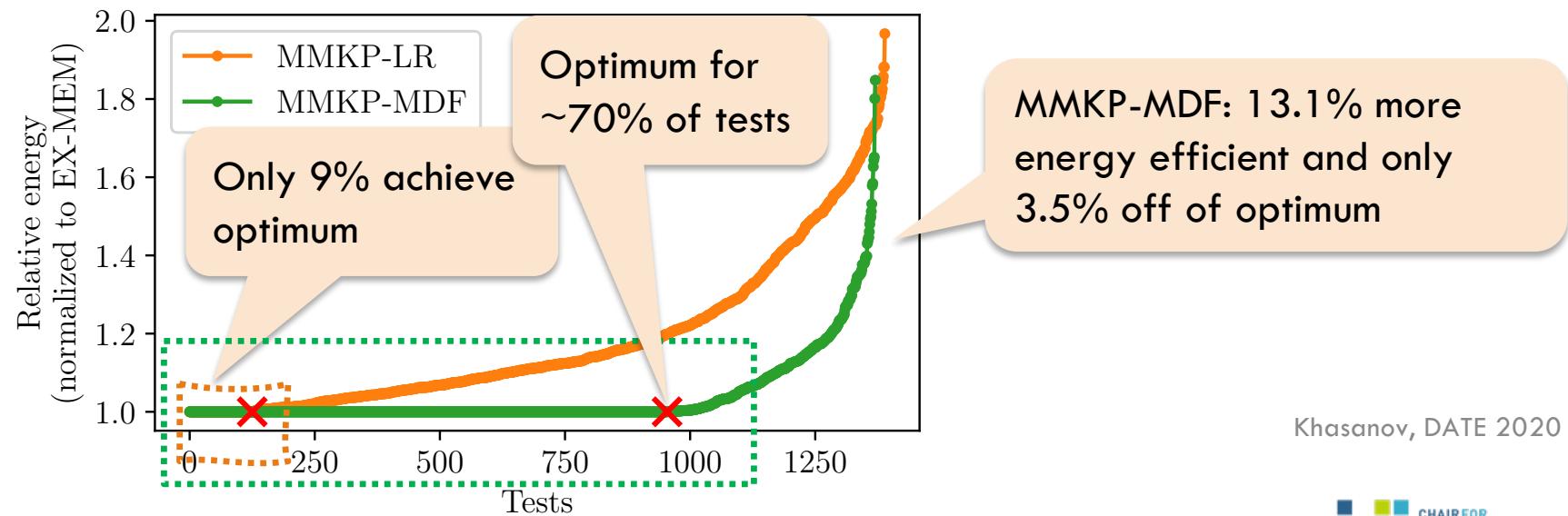
# Run-time adaptability: Tetris

- ❑ Runtime manager aware of symmetries
- ❑ Transform mappings according to resources
- ❑ Less time and energy variations  
(Linux running multiple instances of audio filter applications)



# Run-time adaptability: Run-time (re-)mapping

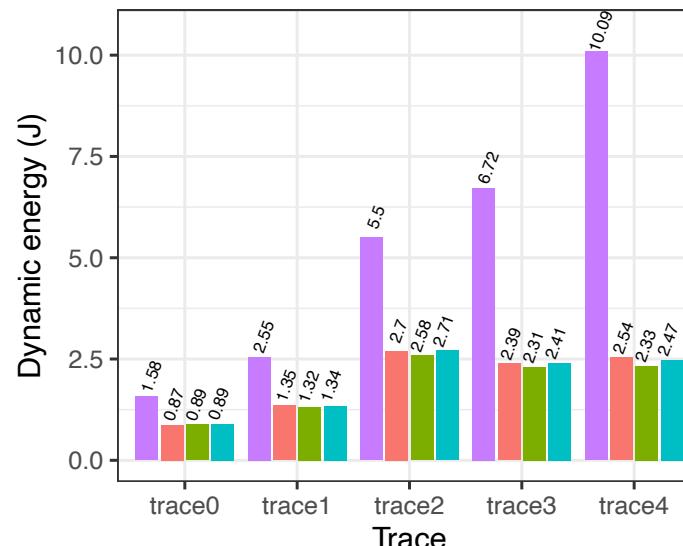
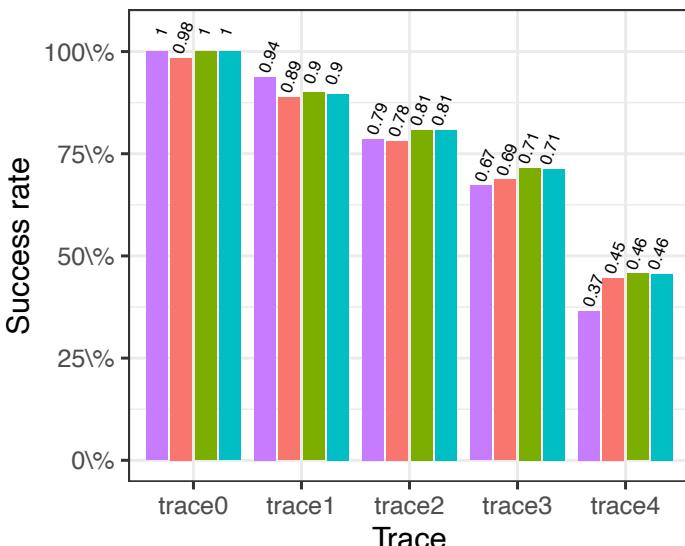
- Fast heuristics: Multiple-choice Multi-dimensional Knapsack Problem (MMKP)
  - Energy and time of canonical mapping configurations computed offline
  - Max Difference First (MDF): Select jobs in decreasing order of energy savings
- Synthetic applications with up to 4 tasks on ODROID (exhaustive exploration)



# Stress benchmark: 4/5G scheduling



- ❑ Multiple graphs per subframe: Deadlines of 0.5 and 2.5 ms!
- ❑ Comparing with work-stealing scheduler and variations of MMKP
- ❑ ODROID and ODROID extended with accelerators

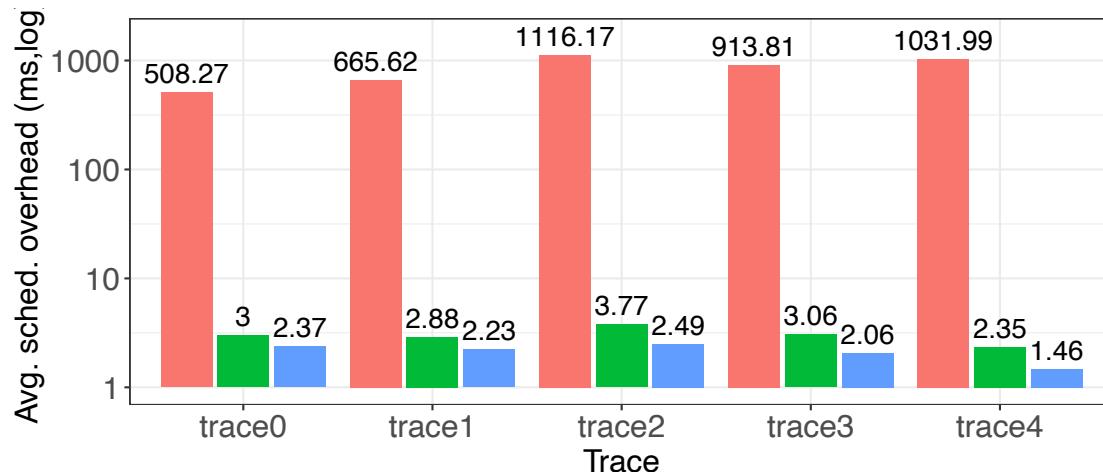


	Work-Stealing	Dynamic scheduler
MMKP-LR		Hybrid, Lagrangian Relaxation
MMKP-MDF		Hybrid, heuristic
MMKP-MDF-J		Hybrid, Knapsack heuristic

Khasanov, ACM TECS 2021

## Stress benchmark: 4/5G scheduling (2)

- ❑ Multiple graphs per subframe: Deadlines of 0.5 and 2.5 ms!
- ❑ Runtime overhead of prototype **Python** algorithms

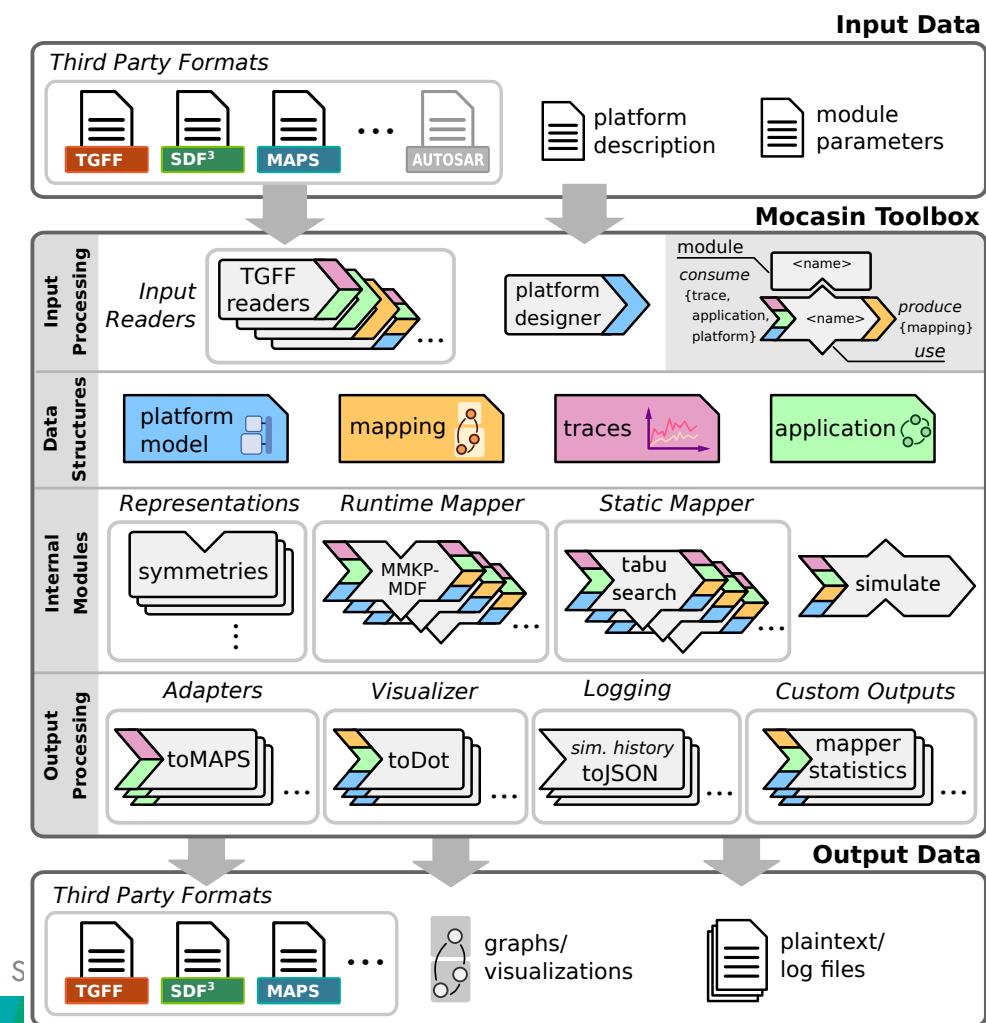
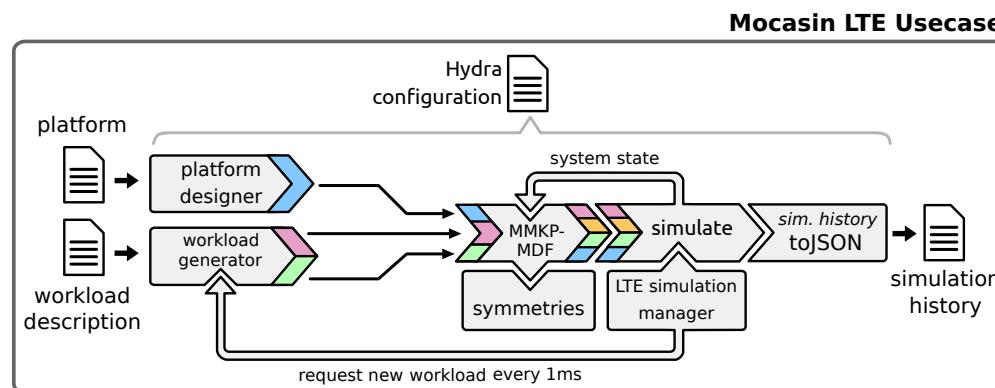


	MMKP-LR	Hybrid, Lagrangian Relaxation
	MMKP-MDF	Hybrid, heuristic
	MMKP-MDF-J	Hybrid, Knapsack heuristic

Khasanov, ACM TECS 2021

# Simulation and tool prototyping

- A python-based exploration framework
  - Tool interfacing
  - Platform generation (w/ calibrated models), heuristics, representations, ...
  - Trace-driven simulation, visualization
  - Tool composition



## In this talk



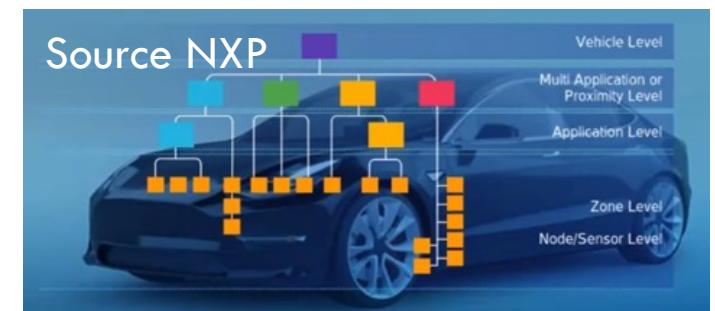
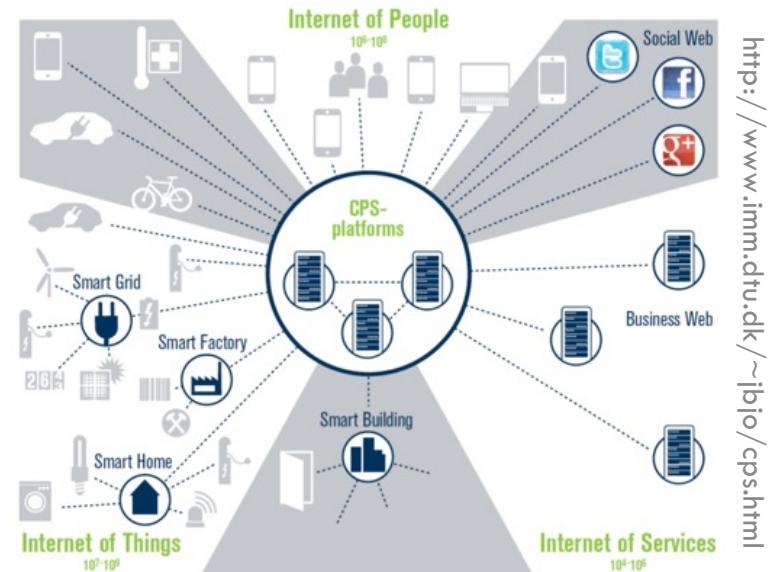
- Introduction
- Quick background
- Runtime adaptivity
- **CPS it's about time**
- Summary

# Reactors: Time-deterministic programming

- CPSs go beyond adaptive dataflow
  - Reactive behavior to external events
  - Time: crucial in the interaction with the physical
  - Distributed and heterogeneous
  - Safety criticality
- Reactors (collab. w/ UC Berkeley and others)
  - Timed semantics + dataflow
  - Allow reasoning about event ordering (determinism)

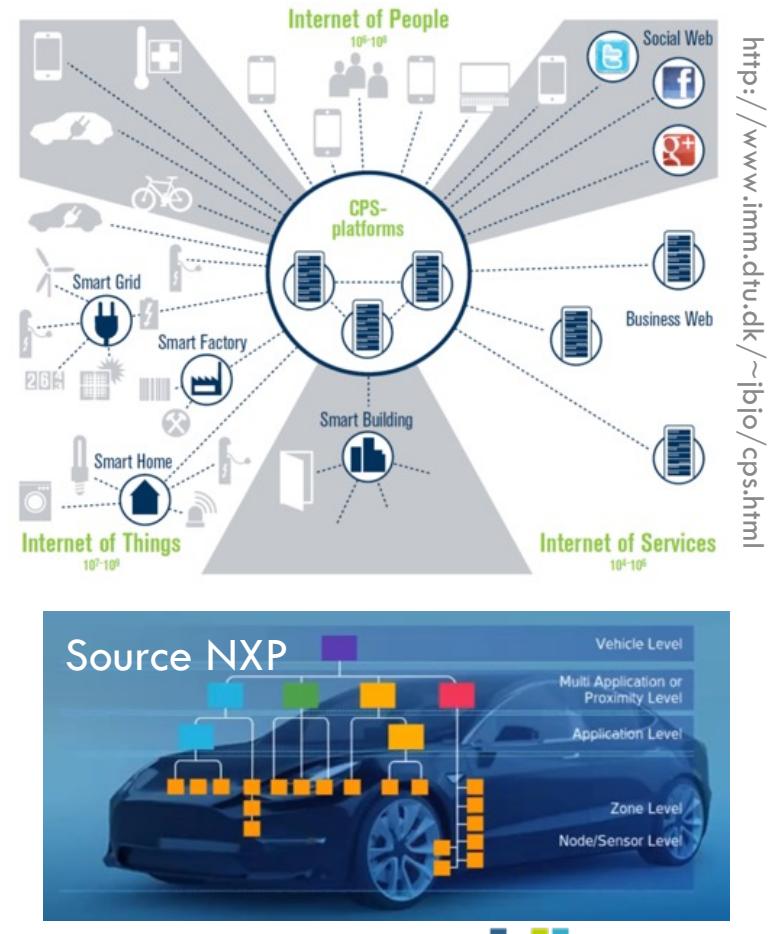
See also Prof. Lee's talk:

[https://www.youtube.com/watch?v=\\_jbdWky4Iys](https://www.youtube.com/watch?v=_jbdWky4Iys)



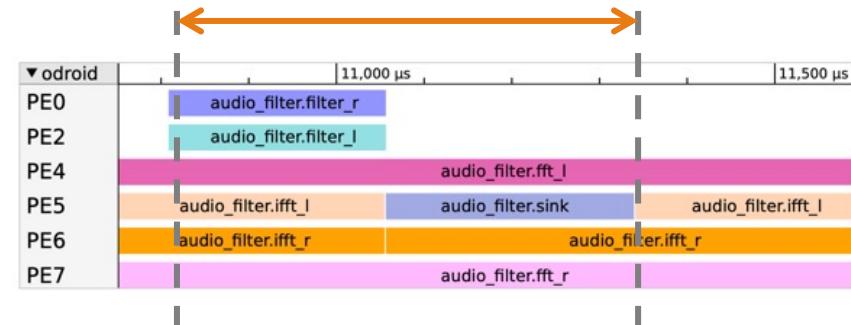
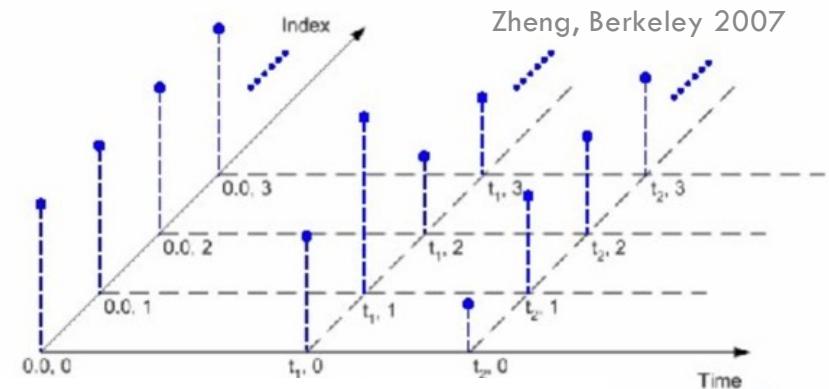
# Reactors: Time-deterministic programming

- Determinism: Given same initial state and inputs, behavior is unambiguously defined
  - Easier debugging and testing
  - Simulations are more representative
  - More tractable analysis and verification
- “Platforms”, e.g., ROS2 or Adaptive AUTOSAR, going asynchronous (publish-subscribe, SOA, actors)



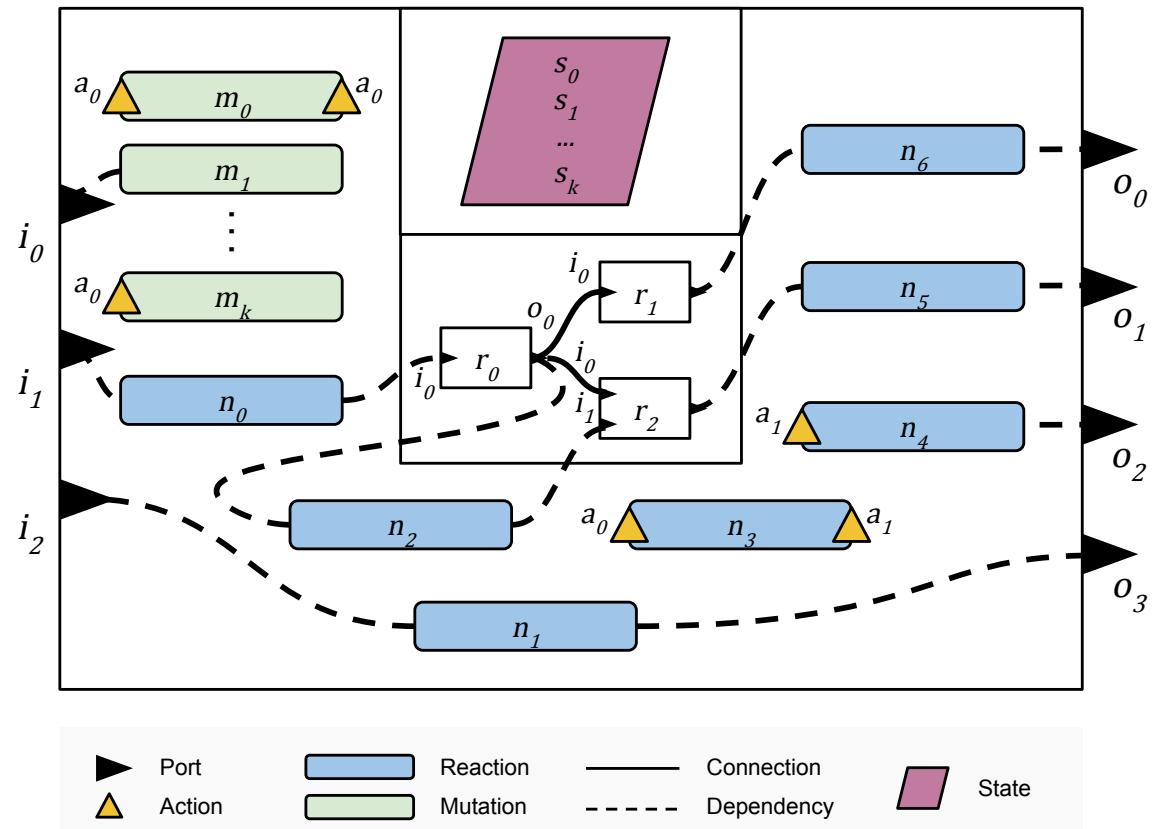
# Logical and physical time

- Logical time (as in synchronous models)
  - Discrete ticks: Execute dataflow graph
  - Absolute simultaneity
  - Total ordering possible (dense time)
- Physical time
  - Continuous, from the environment
  - Simultaneity: Not really
  - Impose **deadlines** on execution
- In reactors: logical time attempts to follow physical time



# The reactor model

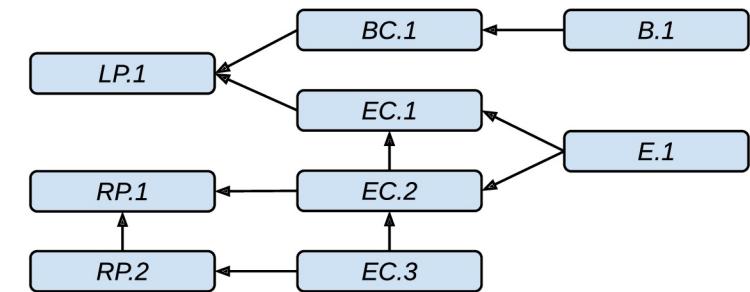
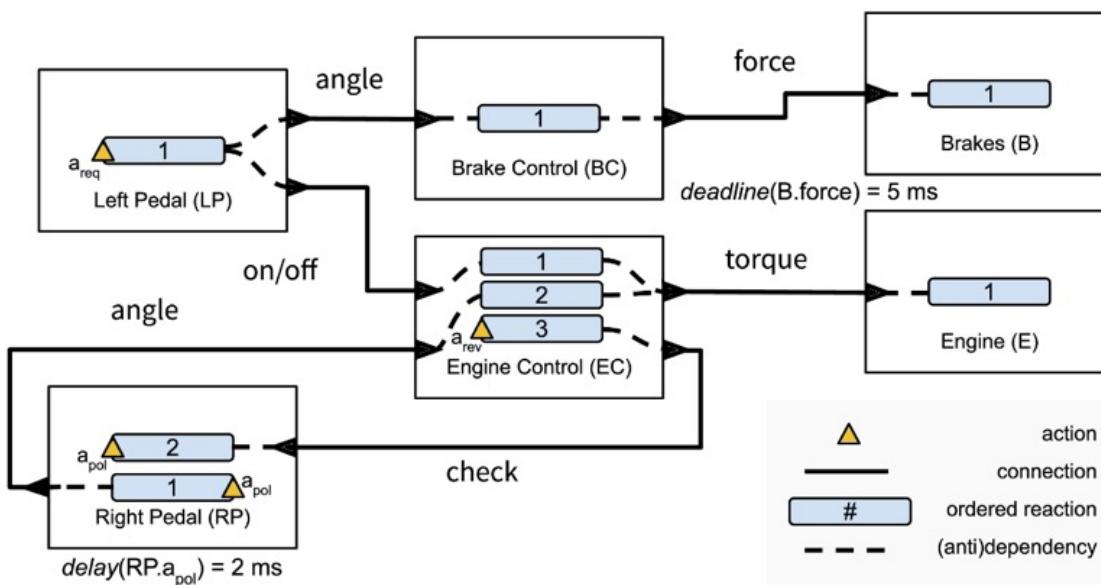
- **Actions** model events (*physical and logical*)
- Behavior in **reactions**, triggered by actions or ports, which can trigger actions
- A **reactor** carries state and contains reactions and reactors
- Reactions have priorities (enforce order)
- **Mutations** (WiP) allow adapting the application



Lohstroh, CyPhy 2019

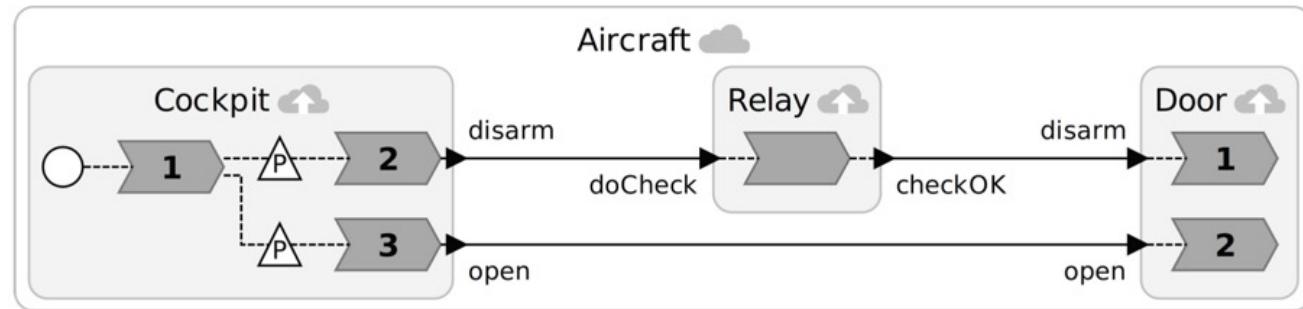
# Reactors and dataflow

- ❑ A reactor program spawns a dependency graph across reactors
- ❑ Example: Power-train control

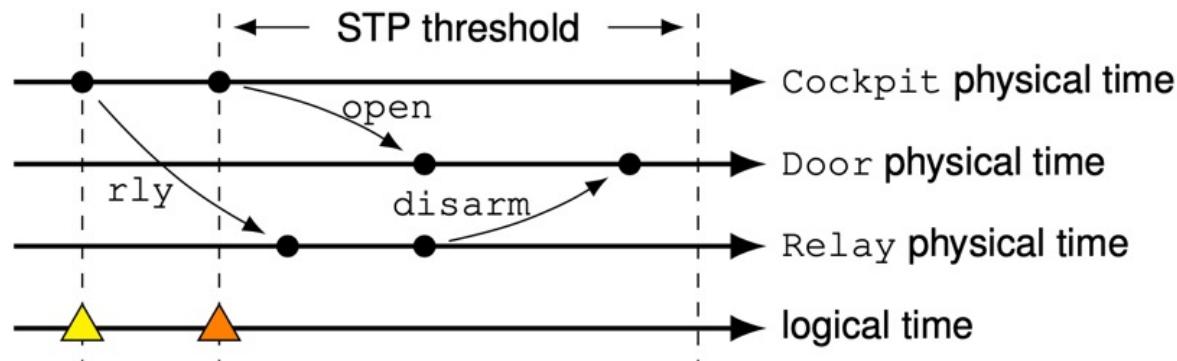


# Distributed execution

- ❑ Order along logical time
- ❑ Leverage physical clocks for synchronization
- ❑ Assumptions



- ❑ Network ports annotated with deadline (D): Relates to time for processing request
- ❑ Safe-to-process: Depends on maximum network latency and clock sync error bounds



Lohstroh, FDL 2020

# Lingua Franca

- Polyglot programming: C, C++, TypeScript, Python, Rust, ...
- IDE using Kieler
  - <https://www.rtsys.informatik.uni-kiel.de/en/archive/kieler/welcome-to-the-kieler-project>
- Code generation and runtime system

The screenshot shows the Kieler IDE interface with two main parts: a code editor and a state transition diagram.

**Code Editor:**

```
target Cpp;

reactor Ping(count:unsigned(40000)) {
    state pings_left:unsigned(count);

    output outPing:void;
    input inPong:void;

    logical action serve;

    reaction(startup, serve) -> outPing {
        pings_left -= 1;
        outPing.set();
    }

    reaction (inPong) -> serve {
        if (pings_left != 0) {
            serve.schedule(10ms);
        }
    }
}

main reactor (count:unsigned(40000)) {
    ping = new Ping(count=count);
    pong = new Pong();

    ping.outPing -> pong.inPing;
    pong.outPong -> ping.inPong;
}
```

**State Transition Diagram:**

```
graph LR
    subgraph PingPong [PingPong]
        direction LR
        P1[Ping] -- "2" --> L(( ))
        L -- "1" --> P2[Pong]
        P1 <-- inPong --> P2
        P2 <-- outPing --> P1
        P2 <-- inPing --> P1
        P2 -- outPong --> P1
    end
```

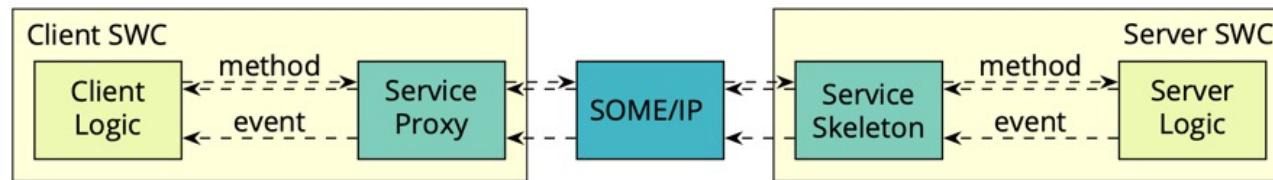
The diagram illustrates a PingPong reactor. It consists of two states: Ping and Pong. Transitions are labeled with actions: '2' from Ping to a loop node (L), '1' from the loop node to Pong, 'outPing' from Ping to Pong, 'inPing' from Pong to Ping, and 'outPong' from Pong back to Ping.

Lohstroh, ACM TECS 2021

# Automotive use case

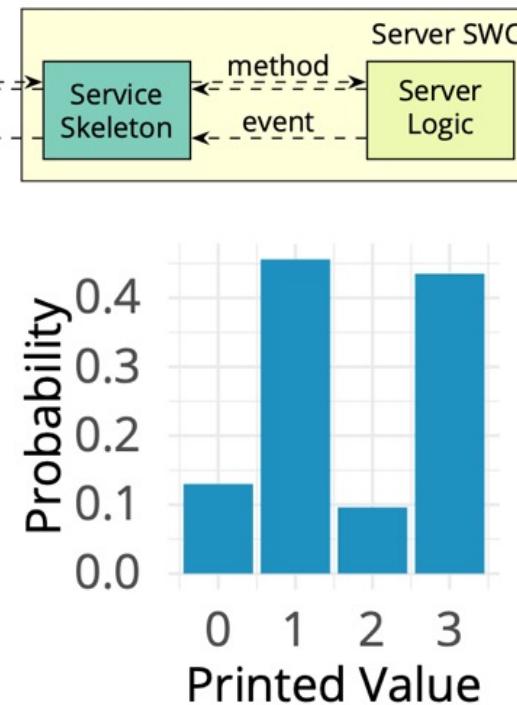


- ❑ Adaptive AUTOSAR: Consortium with most automotive players
- ❑ Service-oriented architecture: C++, futures, concurrency, ...



## Client Code

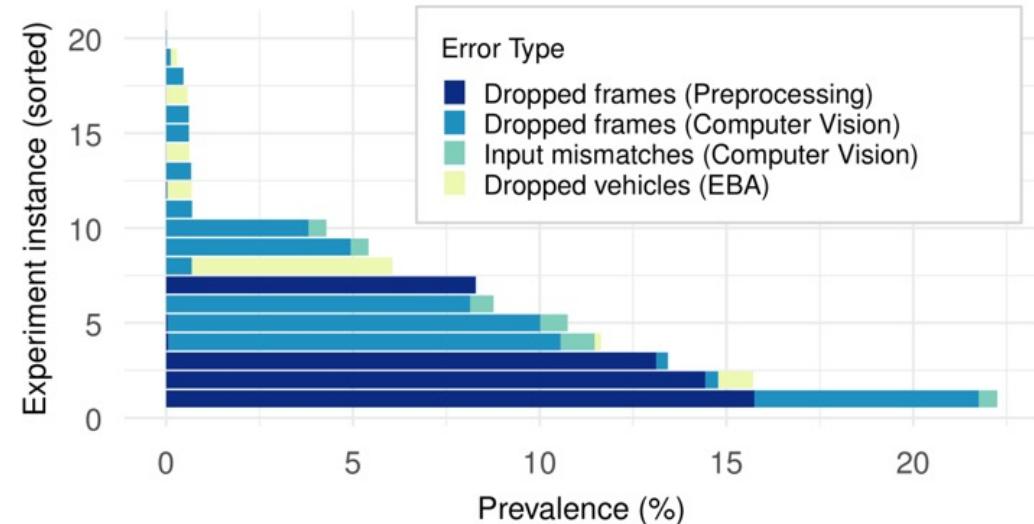
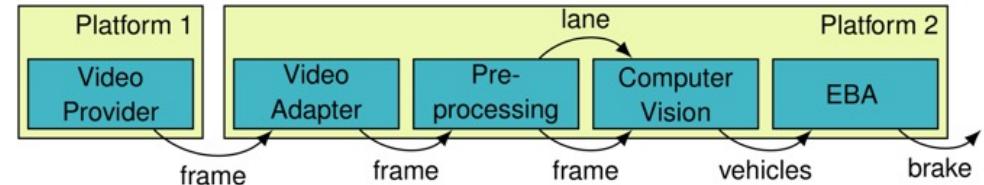
```
1 int main() {  
2     s = ServiceProxy();  
3  
4     s.set_value(1);  
5     s.add(2);  
6     result = s.get_value();  
7  
8     std::cout << result.get();  
9     return 0;  
10 }
```



# Time determinism

- ❑ Emergency break assistant
  - ❑ Test application in AUTOSAR repo
  - ❑ Simple processing pipeline (no buffering)
  - ❑ Hidden assumption: data always read before next item is written
- ❑ Non-determinism leads to different non-deterministic executions!  
(Target system: 2 MinnowBoard Turbot3 boards connected Ethernet)

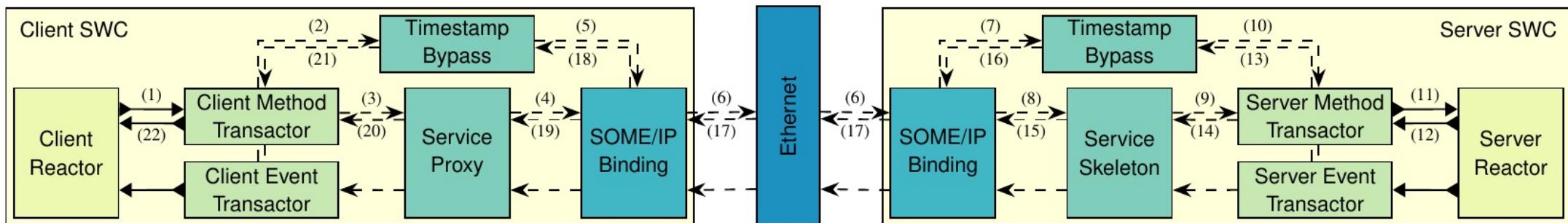
Menard, DATE 2020



# Reactors for automotive software

## □ Reactor integration

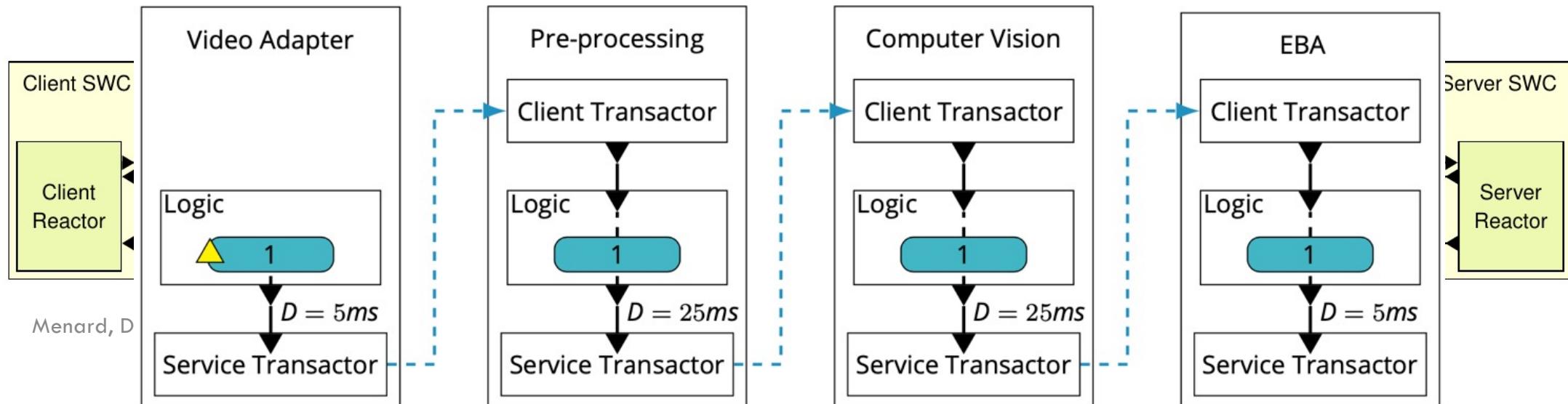
- Transactors convert reactor interfaces to AUTOSAR
- Proxies and skeletons are unmodified (standard compliant)
- Work arounds needed to handle timestamps



Menard, DATE 2020

# Reactors for automotive software

- Time annotations to ensure time-determinism (based on measurements)
  - Work arounds needed to handle timestamps
- Removed all time-bugs with negligible overhead (not trivial to assess)



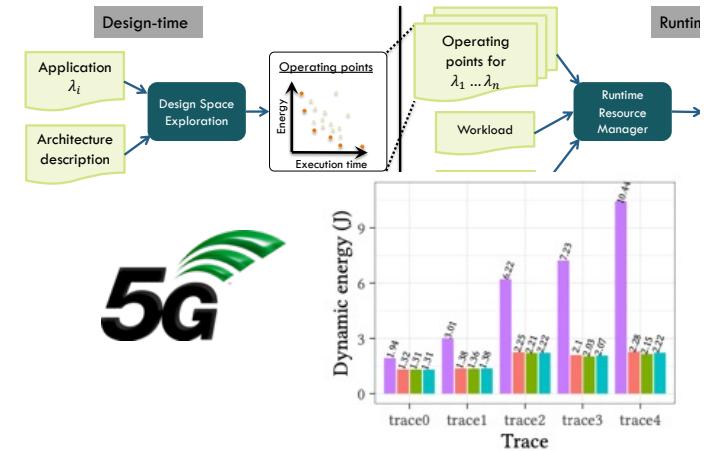
# In this talk



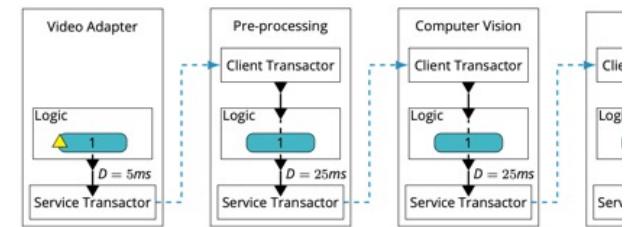
- Introduction
- Quick background
- Runtime adaptivity
- CPS it's about time
- **Summary**

# Summary

- ❑ What can we achieve with formal MoCs?
  - ❑ Improving methods for optimization
  - ❑ Methods for energy-efficient adaptive execution
  - ❑ Adding time for time-determinism in distributed CPS
- ❑ Moving forward
  - ❑ Ongoing work on scaling methods to larger systems
  - ❑ Support for emerging architectures and interconnect
  - ❑ Optimization for reactor programs
  - ❑ Mutations in reactors
  - ❑ ...



5G



# Thanks! & Acknowledgements



Hasna  
Bouraoui      Alexander  
Brauckmann

Karl  
Friebel

Fazal  
Hameed

Asif Ali  
Khan      Robert  
Khasanov



Nesrine  
Khouzami

Galina  
Kozyreva

Christian  
Menard

Julian  
Robledo

Lars  
Schütze      Felix  
Wittwer

..., and previous members of the group (Andres Goens, Sebastian Ertel), and collaborators (E. Lee, M. Lohstroh, H. Härtig, G. Fettweis, A. Kumar)

Funded by



TraceSymm (Number 366764507)



Bundesministerium  
für Bildung  
und Forschung

E4C (Number 16ME0189)



STAATSMINISTERIUM  
FÜR WISSENSCHAFT  
KULTUR UND TOURISMUS



Freistaat  
**SACHSEN**



# References

- Ziegenbein, Dirk, et al. "Future automotive HW/SW platform design (Dagstuhl Seminar 19502)." *Dagstuhl Reports*. Vol. 9. No. 12. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- J. Castrillon, R. Leupers, "Programming Heterogeneous MPSoCs: Tool Flows to Close the Software Productivity Gap", Springer, pp. 258, 2014
- A. Goens, et al. "On the Representation of Mappings to Multicores", Proceedings of the IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC-18), pp. 184–191, Vietnam National University, Hanoi, Vietnam, Sep 2018.
- A. Goens, J. Castrillon, "Embeddings of Task Mappings to Multicore Systems", Proceedings of the IEEE International Conference on Embedded Computer Systems Architectures Modeling and Simulation (SAMOS), Springer, Cham, Jul 2021
- Andres W. Goens Jokisch, "Improving Model-Based Software Synthesis: A Focus on Mathematical Structures", PhD thesis, TU Dresden, 172 pp., May 2021.
- A. Goens, S. Siccha, J. Castrillon, "Symmetry in Software Synthesis", In ACM Transactions on Architecture and Code Optimization (TACO), ACM, vol. 14, no. 2, pp. 20:1–20:26, New York, NY, USA, Jul 2017.
- A. Goens, T. Nicolai, J. Castrillon, "mpsym: Improving Design-Space Exploration of Clustered Manycores with Arbitrary Topologies", In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), IEEE Press, Jul 2021
- R. Khasanov, et al. "Domain-specific hybrid mapping for energy-efficient baseband processing in wireless networks", In ACM Transactions on Embedded Computing Systems (TECS). Special issue of the International Conference on Compilers, Architecture, and Synthesis of Embedded Systems (CASES), Association for Computing Machinery, vol. 20, no. 5s, New York, NY, USA, Sep 2021
- A. Goens, et al. "TETRIS: a Multi-Application Run-Time System for Predictable Execution of Static Mappings", Proceedings of the 20th International Workshop on Software and Compilers for Embedded Systems (SCOPES'17), ACM, pp. 11–20, New York, NY, USA, Jun 2017
- R. Khasanov, J. Castrillon, "Energy-efficient Runtime Resource Management for Adaptable Multi-application Mapping", Proceedings of the 2020 Design, Automation and Test in Europe Conference (DATE), IEEE, pp. 909–914, Mar 2020
- C. Menard, et al. "Mocasin—Rapid Prototyping of Rapid Prototyping Tools: A Framework for Exploring New Approaches in Mapping Software to Heterogeneous Multi-cores", Proceedings of the 2021 Drone Systems Engineering and Rapid Simulation and Performance Evaluation: Methods and Tools, ACM, pp. 66–73, New York, NY, USA, Jan 2021
- M. Lohstroh, et al. "Toward a Lingua Franca for Deterministic Concurrent Systems", In ACM Transactions on Embedded Computing Systems, Association for Computing Machinery (ACM), vol. 20, no. 4, pp. 1–27, May 2021
- M.. Lohstroh, et al. "A Language for Deterministic Coordination Across Multiple Timelines", In Proceeding: 2020 Forum for Specification and Design Languages (FDL), pp. 1–8, Sep 2020
- C. Menard, et al. "Achieving Determinism in Adaptive AUTOSAR", Proceedings of the 2020 Design, Automation and Test in Europe Conference (DATE), IEEE, pp. 822–827, Mar 2020
- M. Lohstroh, et al., "Reactors: A Deterministic Model for Composable Reactive Systems", Cyber Physical Systems. Model-Based Design – Proceedings of the 9th Workshop on Design, Modeling and Evaluation of Cyber Physical Systems (CyPhy 2019) , 2019