VHPC 2023

May 25th, 2023, Hamburg

"Virtualization and Power Optimization of Embedded HPC Applications in AMPERE"

Tommaso Cucinotta

Real-Time Systems Laboratory (RETIS)

ISTITUTO DI TECNOLOGIE DELLA COMUNICAZIONE, DELL'INFORMAZIONE E DELLA PERCEZIONE

Scuola Superiore Sant'Anna

.AMPERE

A Model-driven development framework for highly Parallel and EneRgy-Efficient computation supporting multi-criteria optimisation

.AMPERE

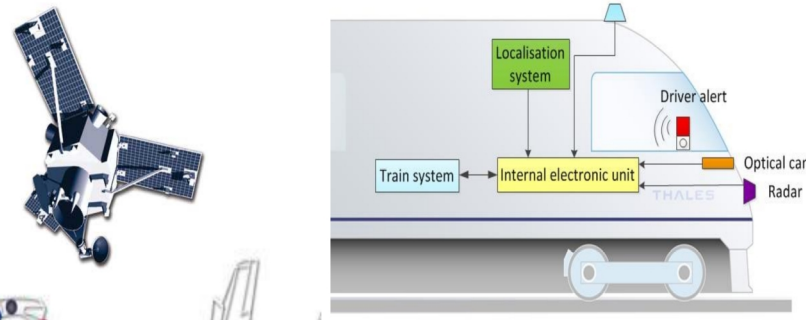
A Model-driven development framework for highly Parallel and EneRgy-Efficient computation supporting multi-criteria optimisation

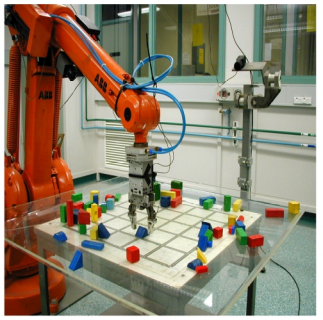
Model-based engineering of high-performance embedded applications on heterogeneous hardware with real-time constraints and energy efficiency

Tommaso Cucinotta – Scuola Superiore Sant'Anna, Pisa (Italy)

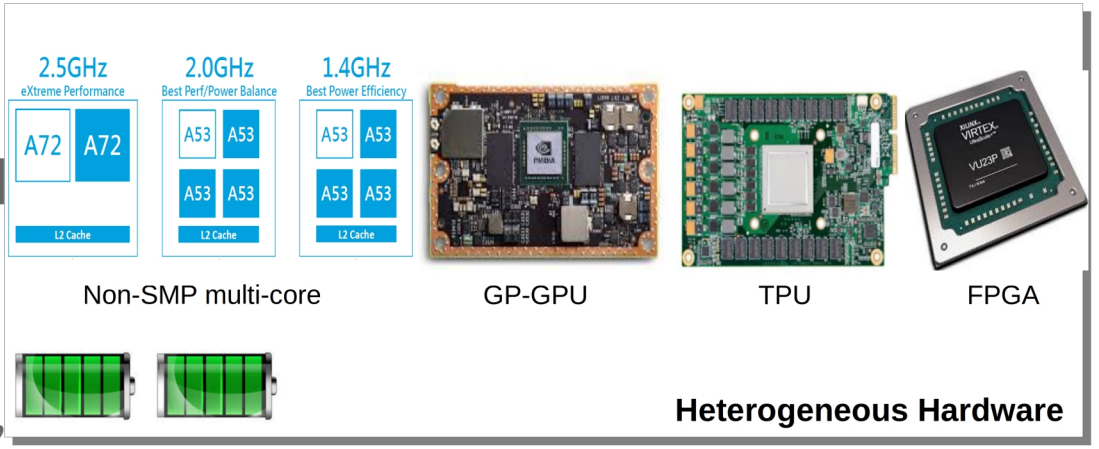
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871669

Introduction & Motivations



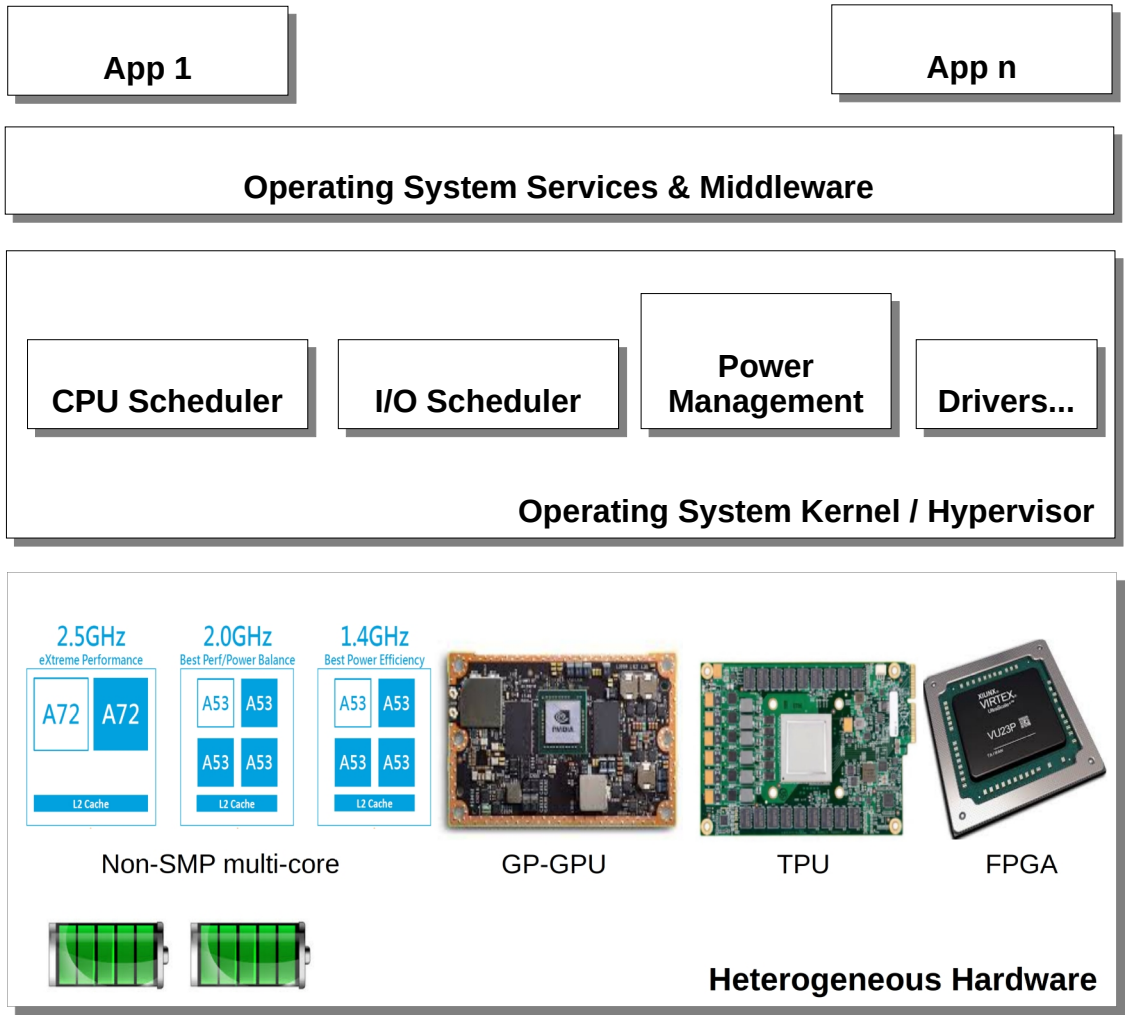


- **CPSs have higher and higher computational performance & reliability requirements**
- **Use of increasingly heterogeneous & interconnected, battery-operated platforms**
  - non-SMP multi-core
  - GP-GPU/TPU acceleration
  - FPGA
- **Heterogeneous platforms needed in soft and hard real-time use-cases**
  - automotive, railways, aerospace, robotics, gaming, multimedia, ...



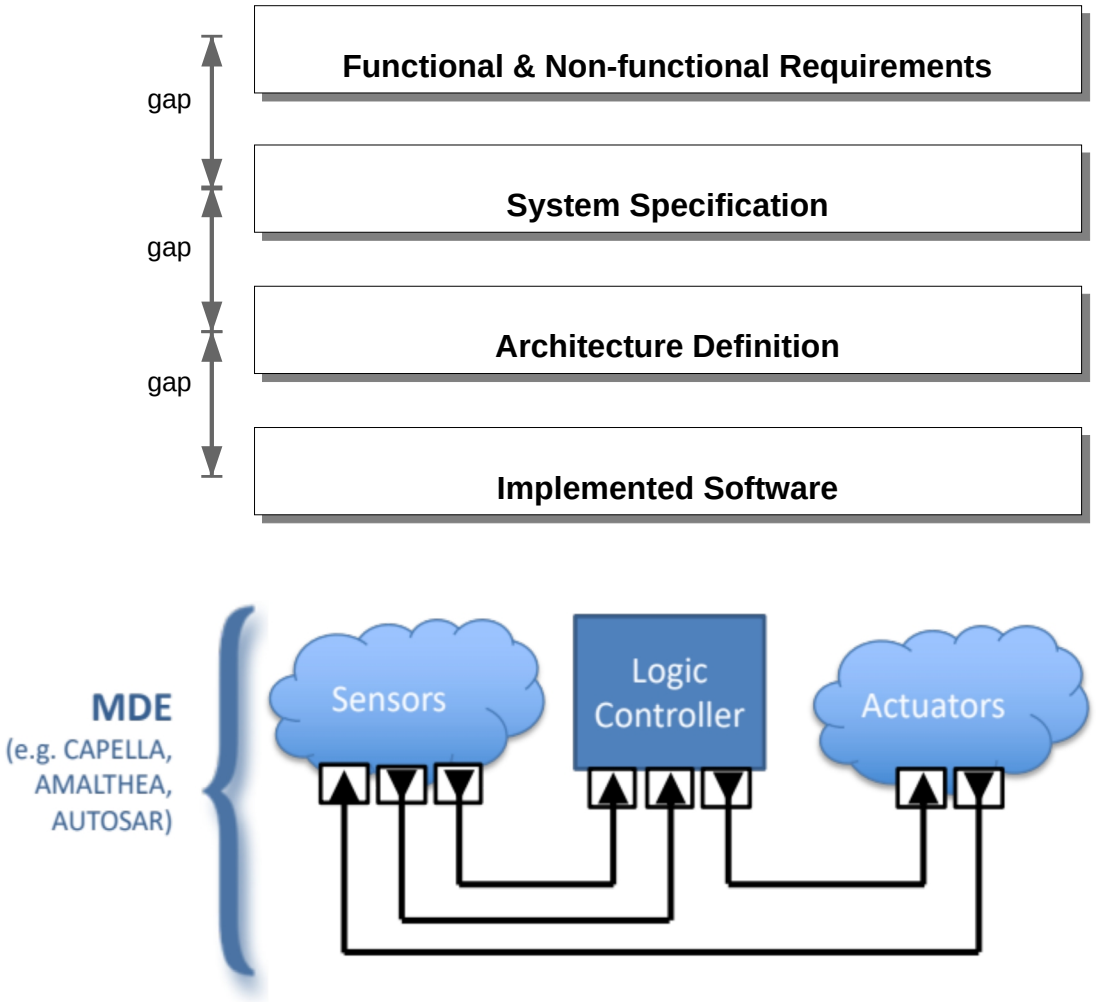
Problems & Challenges

- **Development of software for CPSs is cumbersome!**
  - Optimum usage of underlying hardware parallelism & acceleration
  - Performance vs energy consumption trade-offs
  - Real-time constraints
  - Safety & certification



MDE & Formalisms in Embedded System Design

- **Model-Driven Engineering (MDE)**
  - Fill the gap between high-level specifications and actual system behavior
- **MDE embraces**
  - Formal specification language(s)
  - Model transformation engine(s)
  - Model refinements & composability
  - Automatic code generator(s)
  - Model verifiability
- **=> Correctness-by-construction**

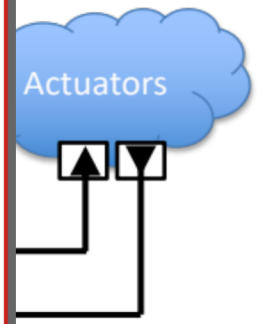


MDE & Formalisms in Embedded System Design

- Model-D
  - Fill th
    high-
    actua
- MDE en
  - Form
  - Mod
  - Mod
    comp
  - Auto
  - Mod
- => Correctness-by-construction

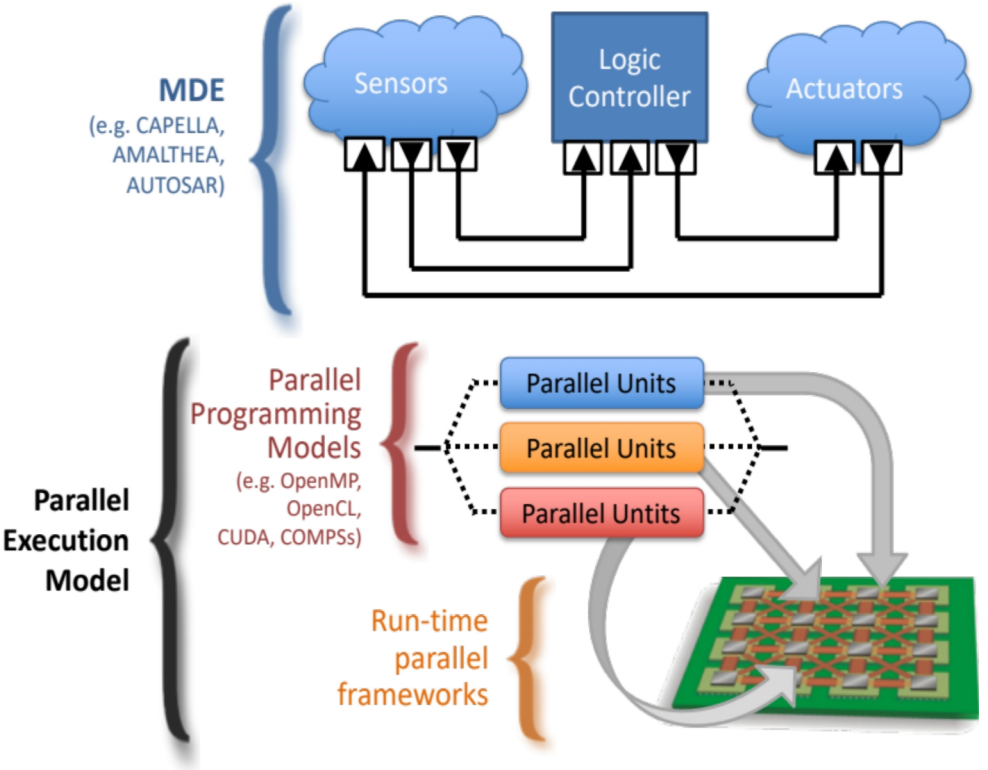
Traditional MDE limitations

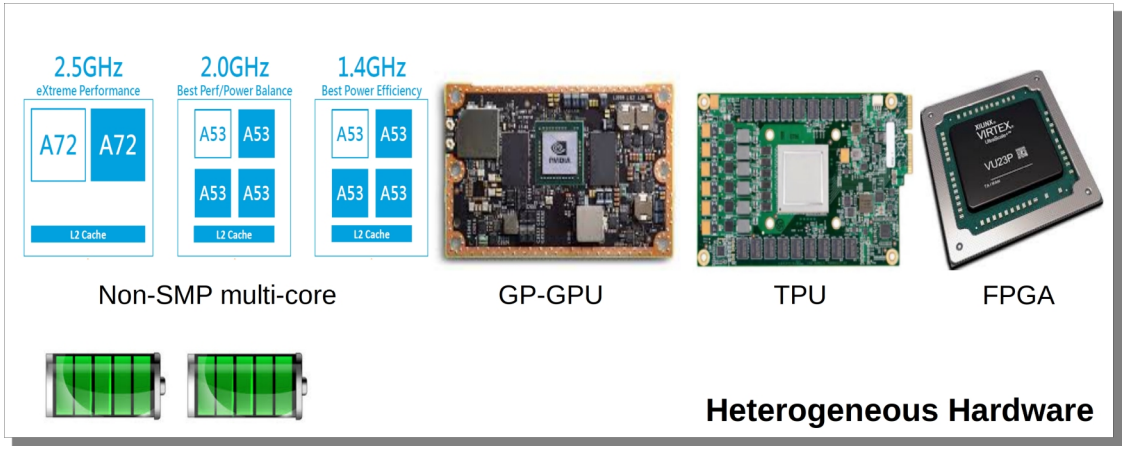
- Single-processor systems or very limited support for multi-core systems
- Struggles at coping with nowadays complex heterogeneous embedded boards



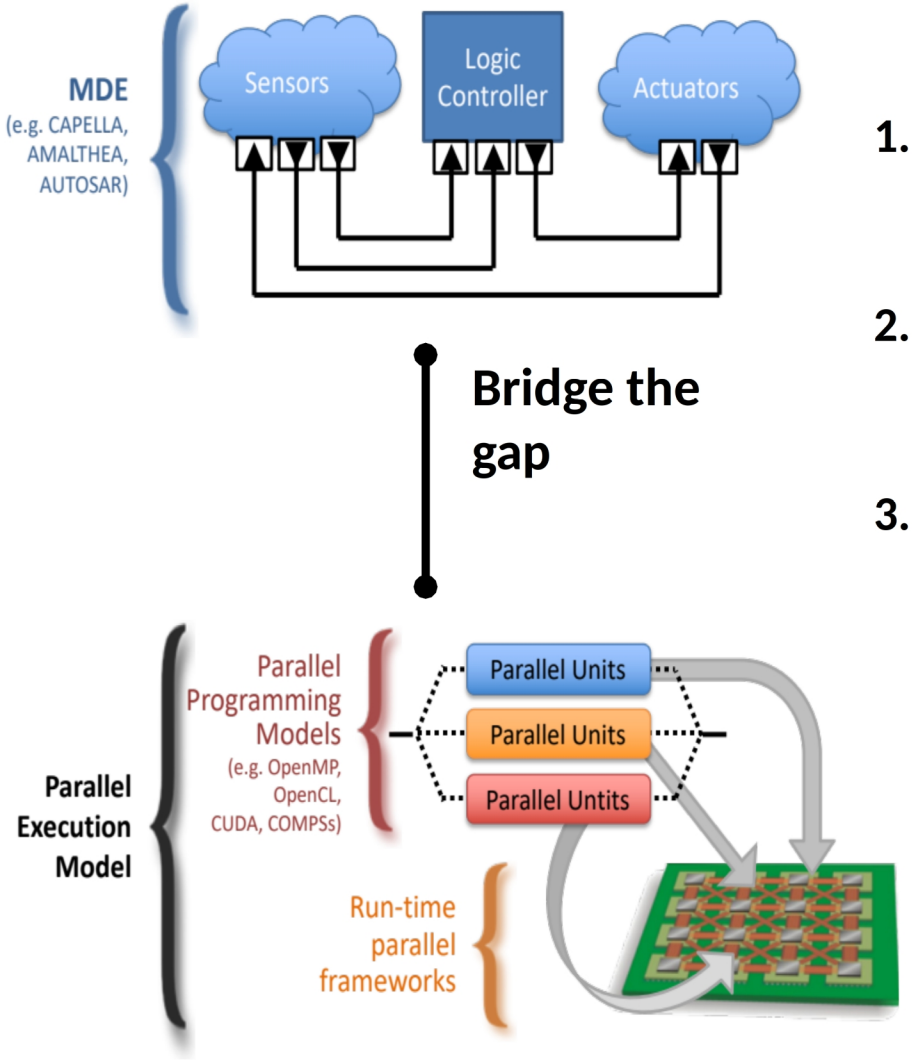
AMPERE Project Goal

- **Fill the gap between**
  - MDE techniques with no/limited parallelism support
  - Parallel-programming models with efficient HW offloading (OpenMP, CUDA, …)
  - Heterogeneity in hardware
- **In presence of non-functional requirements**
  - High-Performance
  - Real-Time Constraints
  - Energy Efficiency
  - Fault Tolerance



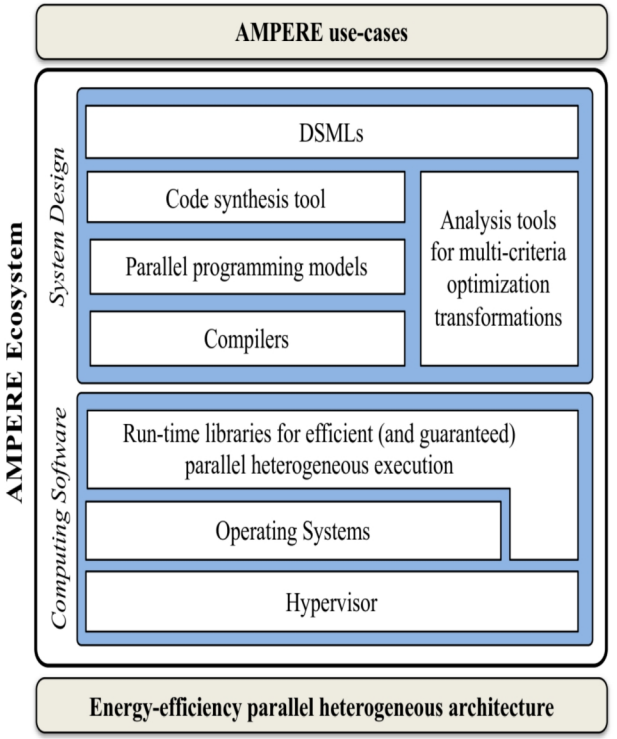


AMPERE Vision



1. **Synthesis methods** for an efficient generation of parallel source code, while keeping non-functional and composability guarantees
2. **Run-time parallel frameworks** that guarantee system correctness and exploit the performance capabilities of parallel architectures
3. **Integration** of parallel frameworks into MDE frameworks

AMPERE Software Development Workflow Overview



System description
- Components/communication
- Functional/NFP
- Etc.

Platform description
- Accel. devices
- Cores/clusters
- Memory model
- Etc.

Model

Meta MDE abstraction

Code Synthesis + Multi-criteria Optimization
- Performance
- Time-predictability
- Energy-efficiency
- Resiliency

Meta PPM abstraction

Compiler
(Correctness + Refined Parallel Structure)

Parallel code (e.g. OpenMP, CUDA graphs)

Resource Allocation
(i.e., mapping/scheduling)
- Monitoring
- Dynamic resource allocation

Run-time framework + OS + Hypervisor

Execution Profile

AMPERE Use-Cases

Obstacle Detection and Avoidance System (ODAS)

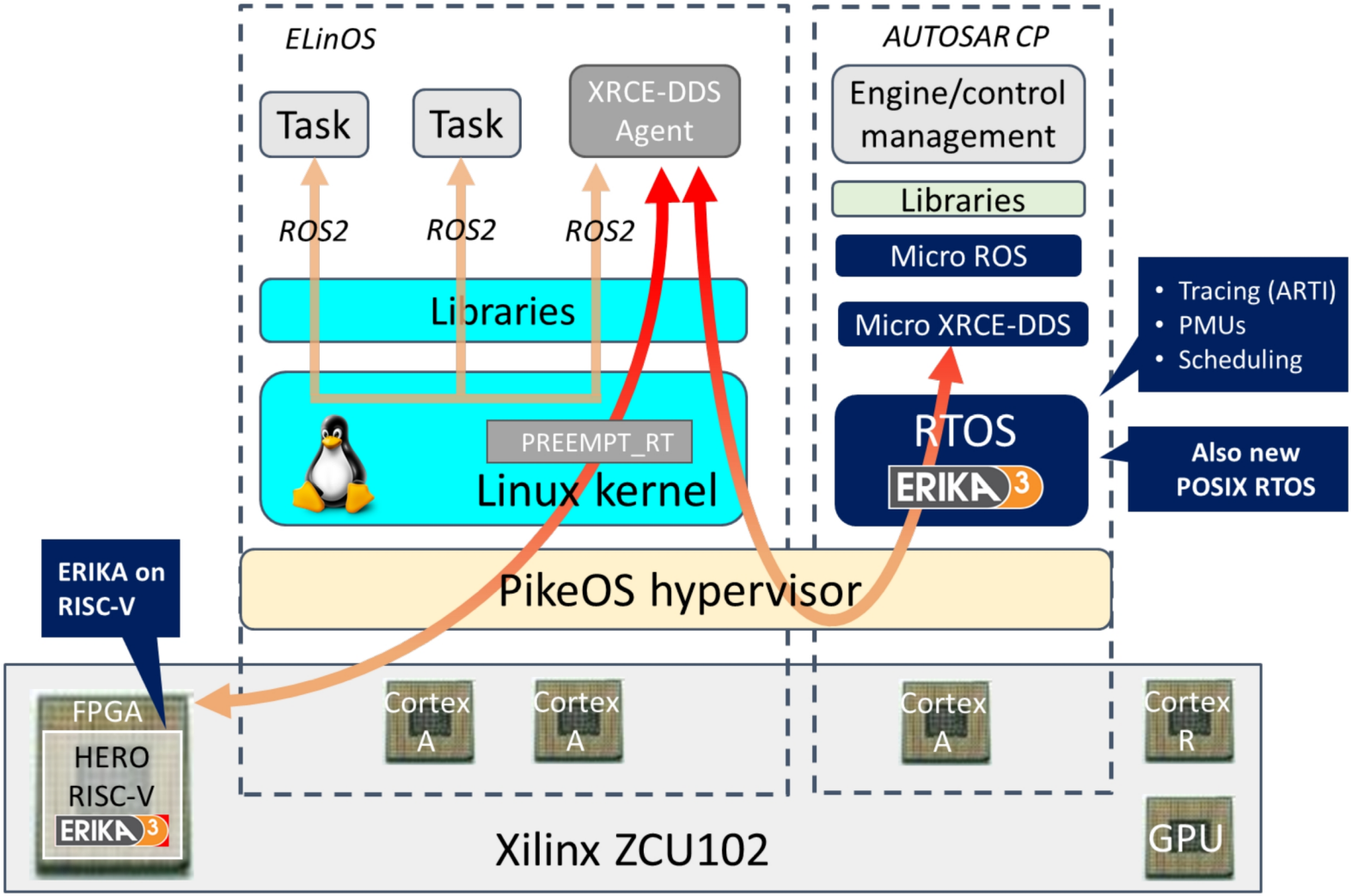
- ADAS functionalities based on data fusion coming from tram vehicle sensors

Predictive Cruise Control (PCC)

- Extends Adaptive Cruise Control (ACC) functionality by calculating the vehicle's future velocity curve using the data from the *electronic horizon*
- Improve fuel efficiency (in cooperation with the powertrain control) by configuring the driving strategy based on data analytics and AI

AMPERE run-time

Reference Architecture (Xilinx board)



Major AMPERE outcomes from SSSA

- **FRED + DART**
  - Automated FPGA floorpolanning and time-scheduled FPGA access at run-time
- **Real-Time Systems Simulator extensions**:
  - Power-aware and thermal-aware simulations for multi-core big.LITTLE platforms
  - Execution times, power consumption and temperature data profiled from real runs on ODROID-XU4, Raspberry PI 4, Xilinx US+
  - big.LITTLE CBS (BL-CBS): simulation only
- **SCHED_DEADLINE enhancements (prototypes)**
  - Adaptive Partitioned EDF (APEDF, w/o DVFS): simulations, implementation & experimental results
  - APEDF with schedutil integration (w DVFS): implementation & experiments (→Gabriele/Luca)
  - APEDF test for migrating SCHED_DEADLINE reservations: theory/simul.
- **Admission tests for APEDF-FF on big.LITTLE**
- **Multi-criteria optimization of real-time DAGs on heterogeneous platforms**
- **Real-Time Framework (RETIF): declarative RT deploym. on multi-cores**

Real-Time Virtualization in AMPERE

- **PikeOS is a certified separation kernel / hypervisor**
  - isolates multiple “partitions” on different CPUs
  - partitions completely isolated, unless they need to interact
  - certified for real-time operation for various industries
    - avionics & space, railway, automotive, industrial automation (DO-178C, EN 50128/50657, ISO 26262, IEC 61508/62304)
  - supports a few guest OSes (PikeOS native, ELinOS)
- **Virtualization in AMPERE**
  - one partition runs hard real-time workloads on the Open Erika AUTOSAR-compliant RTOS from Evidence/Huawei
  - the other partition runs soft real-time and non real-time workloads on Linux

Real-Time Virtualization and FPGA acceleration

- Xilinx UltraScale+ ZCU102 board
  - Arm Cortex A CPU @ 1.2 GHz, FPGA fabric
  - Arm R CPU, Mali GPU (unused in AMPERE)
- Comparing bare-metal vs virtualized FPGA access

	FPGA call w/out reconf.	FPGA call w/reconf.
Bare-metal: PetaLinux 2020.2 Linux kernel 5.4 PREEMPT_RT	915 us	4532 us
Virtualized: PikeOS 5.1, ELinOS 7.1 Linux kernel 5.10 PREEMPT_RT	1063 us	4724 us
Virtualization Overhead (difference)	148 us	191 us

PARTSim / PARTProf

- **Table-based power-consumption curves integrated in RTSim**

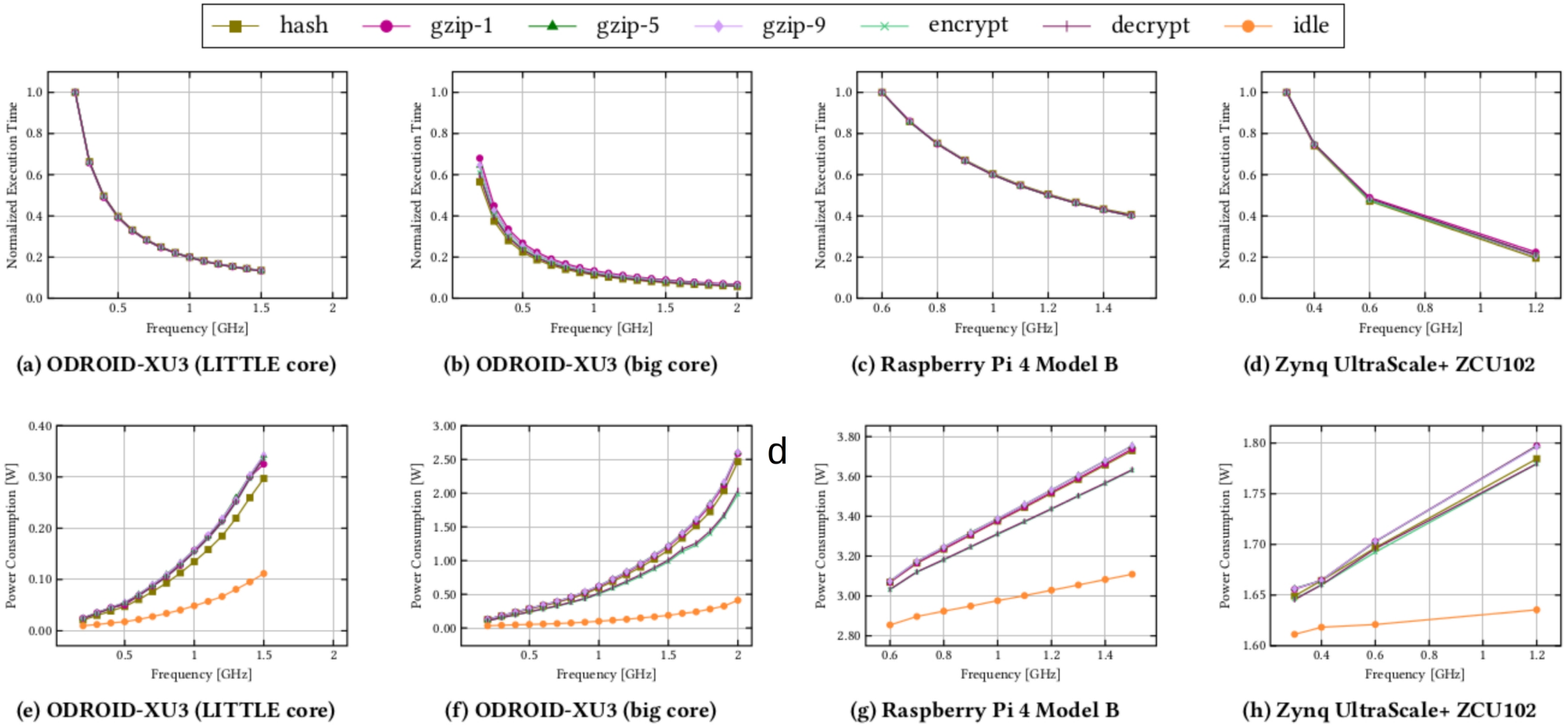


Figure 3: Variation of tasks execution time — (a), (b), (c), (d) — and power consumption — (e), (f), (g), (h) — when varying operating CPU frequency on various embedded platforms and core types. All execution times are normalized with respect to the longest execution time for each workload type, usually registered for the smallest frequency of the least powerful core on each platform. Notice that the methodology applied for power consumption estimation varies from platform to platform.

PARTSim / PARTProf

- **Table-based execution-time curves integrated in RTSim**

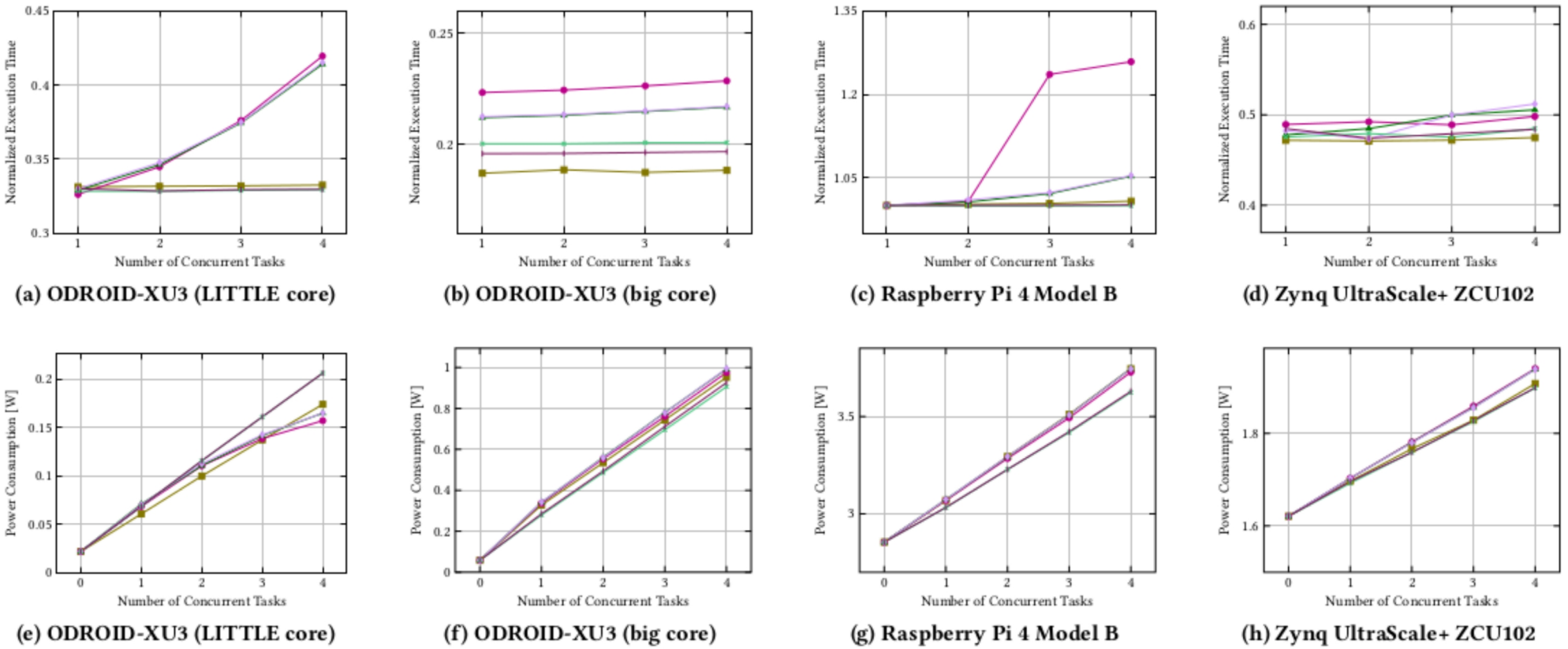


Figure 4: Variation of tasks execution time — (a), (b), (c), (d) — and power consumption — (e), (f), (g), (h) — when varying the number of tasks of the same workload type running on various embedded platforms and core types. Each concurrent task is pinned to a separate core, with frequency fixed at 600 MHz for all platforms. All execution times are normalized with respect to the longest execution time for each workload type, registered for the smallest frequency of the least powerful core on each platform. The value of "0" running tasks indicates the consumption of the target platform/island when no task is running (idle). Notice that the methodology applied for power consumption estimation varies from platform to platform.

Thermal-aware simulations

- Thermal model, experimental data, model fitting

$$\begin{bmatrix} \dot{T}_1(t) \\ \dot{T}_2(t) \end{bmatrix} = \begin{bmatrix} -\frac{\alpha+\beta}{C} & \frac{\beta}{C} \\ \frac{\beta}{C} & -\frac{\alpha+\beta}{C} \end{bmatrix} \begin{bmatrix} T_1(t) \\ T_2(t) \end{bmatrix} + \frac{1}{C} \begin{bmatrix} P_{w1} + \alpha T_e \\ P_{w2} + \alpha T_e \end{bmatrix} \quad (2)$$

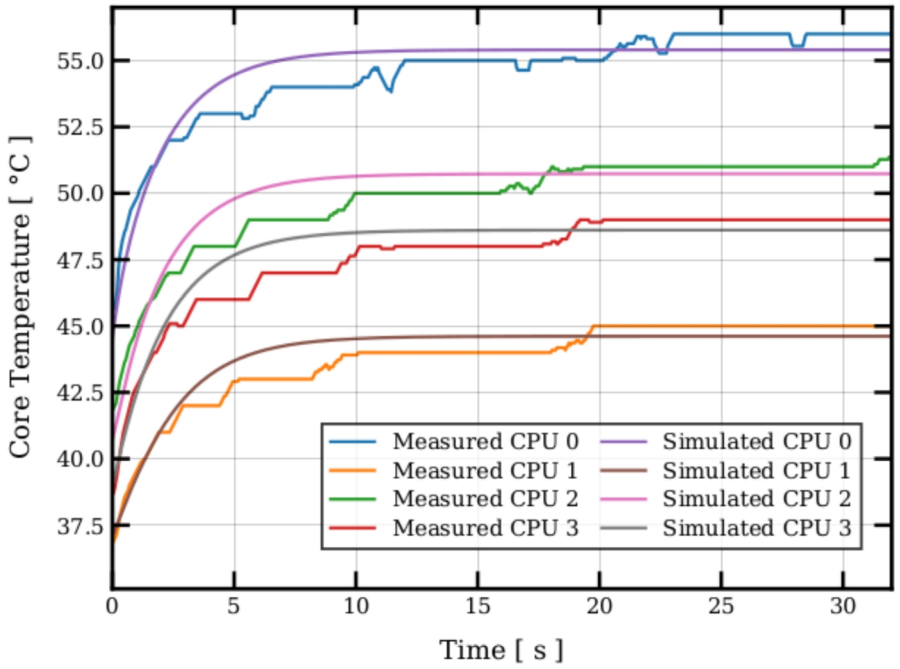


Figure 2: Comparison between an evolution of the temperature of 4 Cortex-A15 (big) CPU cores and the simulated evolution by our work in progress model for the same scenario.

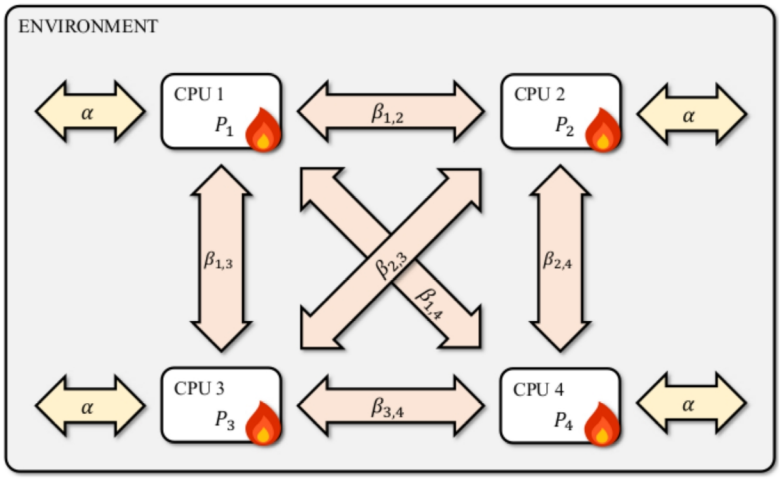


Figure 1: Graphical representation of all heat transfers across CPUs on a 4-core CPU island and the environment in our thermal model.

$$T(t) = \left(\int_{t_0}^{t} e^{A(t-\tau)} d\tau \right) b + e^{A(t-t_0)} T(t_0).$$

BL-CBS

- **BL-CBS**
  - ready to replace RT task at each (re-)activation
  - choosing CPU s.t.:
    - per-CPU schedulability constraint is satisfied (P-EDF)
    - expected power consumption is minimized
  - RT task waking up → need to scale frequency up
  - RT task sleeping → opportunity to scale it down
- on wake-up, BL-CBS chooses **min-power move** among:
  - keep task on CPU/island where it is
  - move it to different CPU on same island, possibly scaling island freq up
  - move it to different island, possibly scaling island freq up

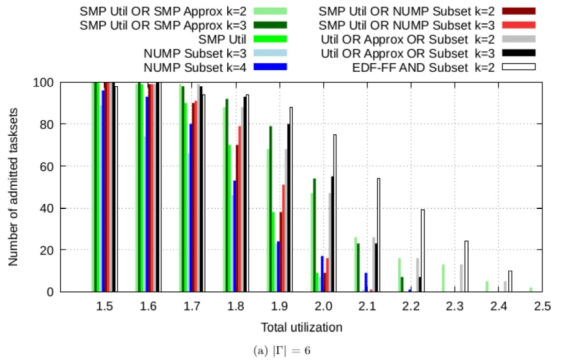
BL-CBS Admission Tests

- How can we admit a set of RT tasks making sure they'll fit into some CPU no matter their sleep/wake-up order?

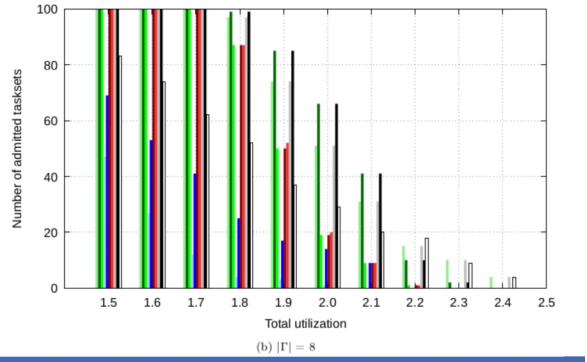
$$\sum_{i=1}^{m} U_i \leq \frac{n\beta + 1}{\beta + 1}$$

- Traditional P-EDF test based on max utilization

$$\beta = \left\lfloor \frac{1}{\alpha} \right\rfloor \text{ and } \alpha = \max_{i=1..m}\{U_i\}.$$



(a) $|\Gamma| = 6$

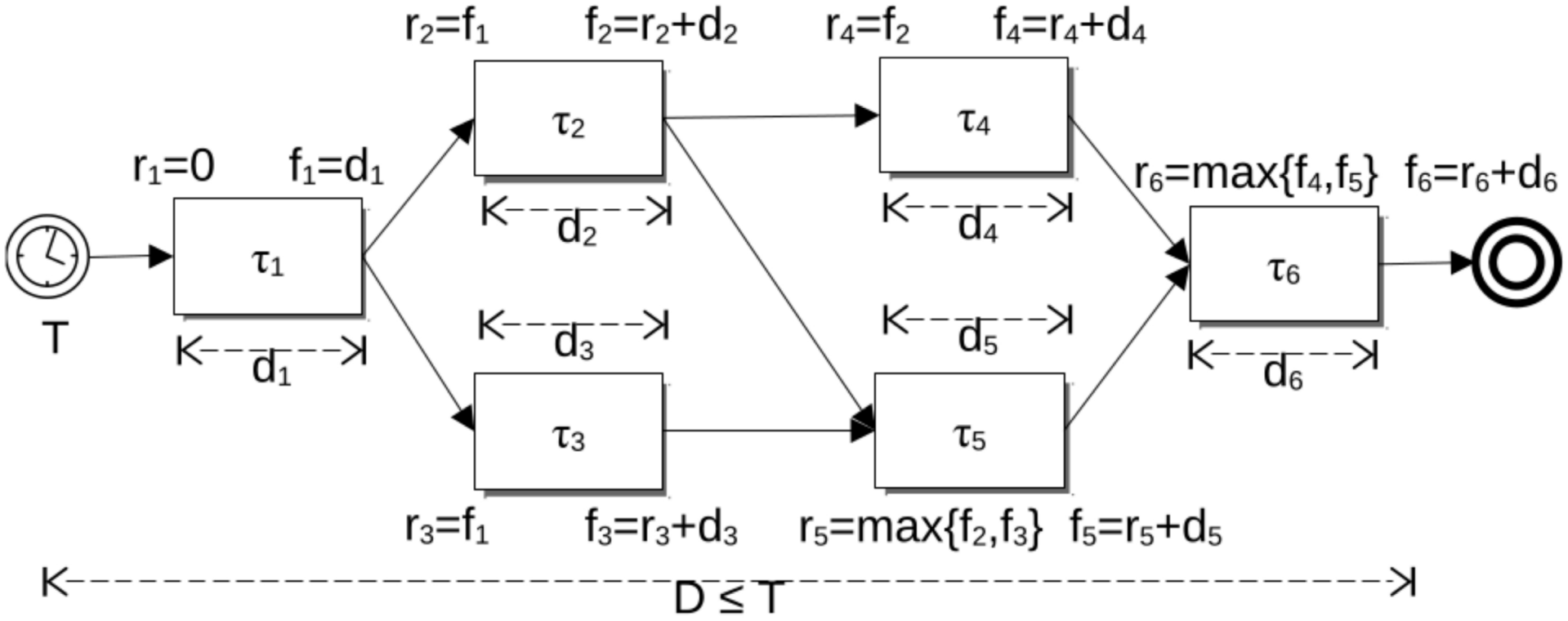


(b) $|\Gamma| = 8$

- Improved tests
  - based on 2nd, 3rd, kth max utilization
  - consider big.LITTLE (max) capacity differences

Multi-criteria optimization

- **Software model**
  - multiple RT DAGs, with known WCETs and AvgETs
  - end-to-end deadline constraint (D)
  - minimum inter-arrival activation times (T)



Multi-criteria optimization

- **Hardware model**
  - multiple islands of cores/PUs under same clock (DVFS)
  - known WCETs, AvgETs and Power Consumption of all tasks on all islands at each frequency;
    - either table-based, or model-based

$$C_{i,s,m} = C_i^{ns} + \frac{C_i - C_i^{ns}}{\xi_s \phi_{s,m}} \phi_{s,ref}$$

  - hardware accelerator islands (GPUs / slots of FPGA), again with known ETs, Powers and ctx-sw delay+pow

MIQCP Optimization

- Mixed-Integer Quadratic Constraint Programming
  - topology constraints
  - end-to-end deadline constraint
  - minimize power
  - or maximize relative slack w power budget
  - max per-DAG utilization
  - per-CPU schedulability constraint ($U^{max} = 0.95$ is tunable)

$$\forall k,\ r_{s_j,k+1} \geq r_{s_j,k} + T_j; \tag{1}$$

$$\forall k,\ f_{e_j,k} \leq r_{s_j,k} + D_j. \tag{2}$$

$$D_j \leq T_j,\ \forall G_j \in G. \tag{3}$$

$$\min \sum_{p \in P} \sum_{m \in \phi_{s(p)}} \left(y_{s(p),m} \mathcal{P}^I_{s(p),m} + \Delta\mathcal{P}_{s(p),m} \sum_{\tau_i \in \Gamma} \frac{z_{i,p,m} C_{i,s(p),m}}{T_{j(i)}} \right) \tag{5}$$

$$\Delta\mathcal{P}_{s,m} = \mathcal{P}^B_{s,m} - \mathcal{P}^I_{s,m} \tag{6}$$

$$\max \min_{G_j \in G} \frac{D_j - f_{e_j}}{D_j} \tag{7}$$

$$\sum_{p \in P} \sum_{m \in \phi_{s(p)}} \left(y_{s(p),m} \mathcal{P}^I_{s(p),m} + \Delta\mathcal{P}_{s(p),m} \sum_{\tau_i \in \Gamma} \frac{z_{i,p,m} C_{i,s(p),m}}{T_{j(i)}} \right) \leq B \tag{8}$$

$$u^{max}_{j,p} = \max_{S \in W_j} \sum_{i \in S} \left(\frac{1}{d_i} \sum_{m \in \Phi_{s(p)}} z_{i,p,m} C_{i,s(p),m} \right) \tag{13}$$

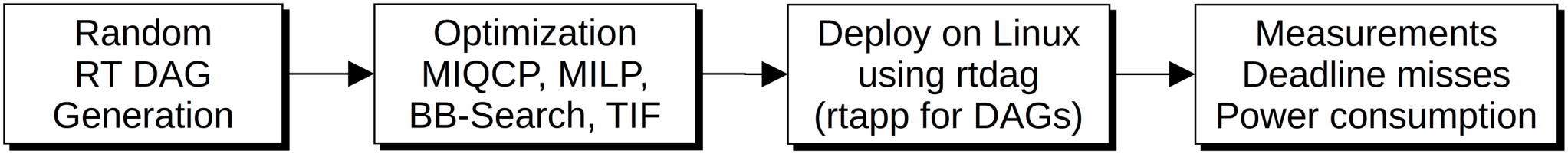
$$\sum_{j=1}^{n_G} u^{max}_{j,p} \leq U^{max}, \tag{14}$$

MIQCP vs Heuristic Solvers

- **MIQCP outputs**
  - optimum task-to-CPU mapping
  - optimum CPU island DVFS configuration (frequency)
  - optimum software-/accelerated-task choices
  - optimum intermediate deadlines (SCHED_DEADLINE)
- **MILP**
  - end-to-end deadline splitting, then BLP
  - suboptimum intermediate deadlines
- **BB-Search**
  - for each task to island mapping
    - end-to-end deadline splitting, then place worst-fit (at max frequency)
    - scale-down to minimum frequency that still fits
  - take the best (minimum power) among all
- **TIF**
  - fit worst-fit into LITTLE as much as possible, then other island(s) (at max frequencies)
  - scale-down islands to minimum frequency that still fits

Experimentation on Linux

- Boards
  - ODROID-XU4 (big.LITTLE)
  - Xilinx UltraScale+ ZCU102 (quad-core A52, with FPGA)
- Kernel modifications
  - Inverted SCHED_DEADLINE and RT classes
  - Disabled SCHED_DEADLINE runtime DVFS adaptation
    - to control it more easily from userspace
- **Methodology**



MIQCP Solver Evaluation
Number of solved problems

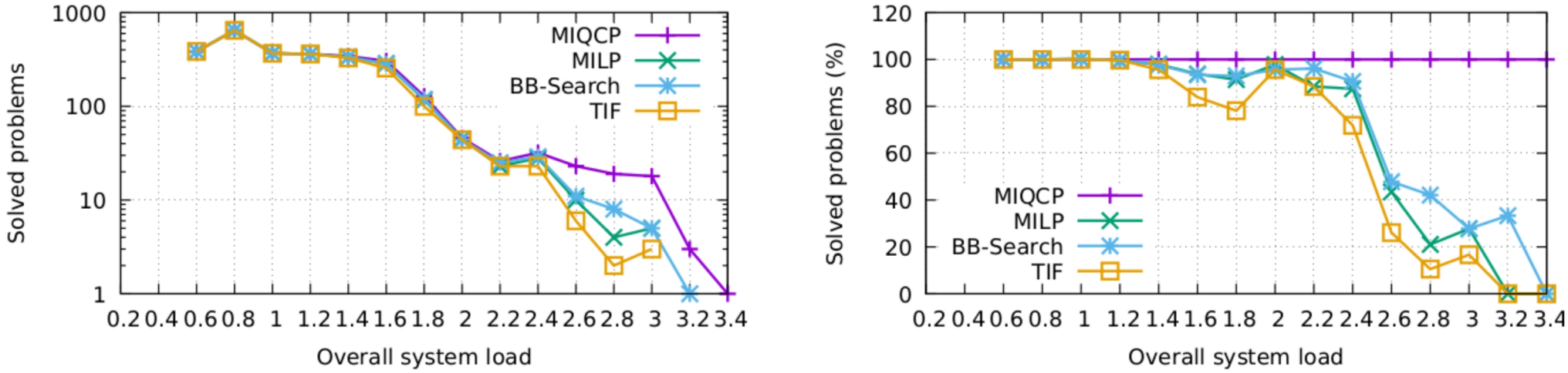


Fig. 3. Left: number of problems (on the Y axis, in logscale) for which various solving strategies (in various curves) found a solution, for various ranges of the overall system unscaled load (on the X axis, in bins of 0.2). Right: same information, in percentage with respect to the problems solved by MIQCP.

MIQCP Solver Evaluation
Minimized power vs optimization time

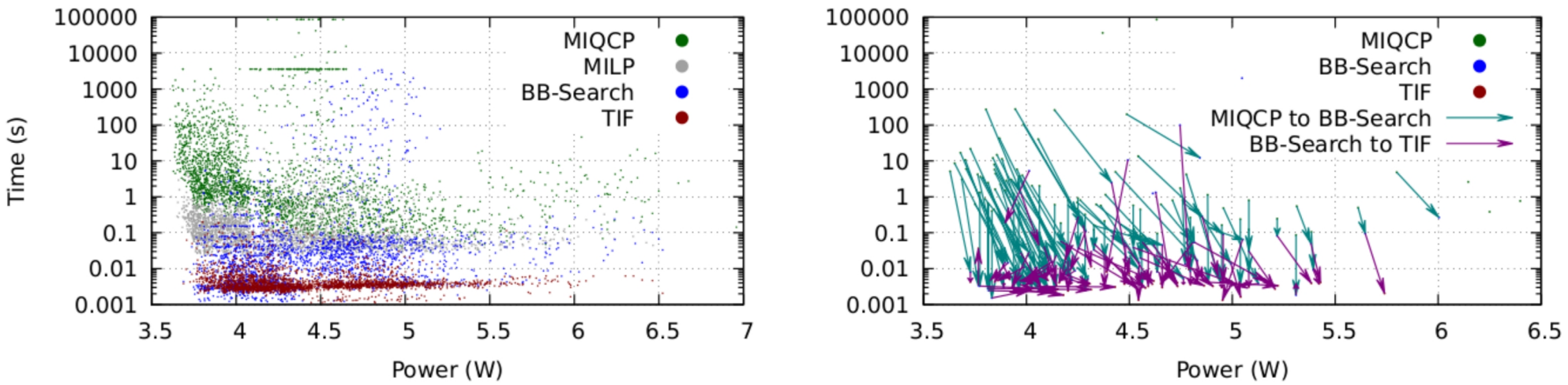


Fig. 5. Left: average solving time (Y axis) obtained by the various optimizers (different data series) vs the achieved optimized average power (X axis). Right: visualization with arrows connecting points obtained by various solvers for the same problems (random subset of 100 problems).

MIQCP Solver Evaluation
Optimization effectiveness at varying system load

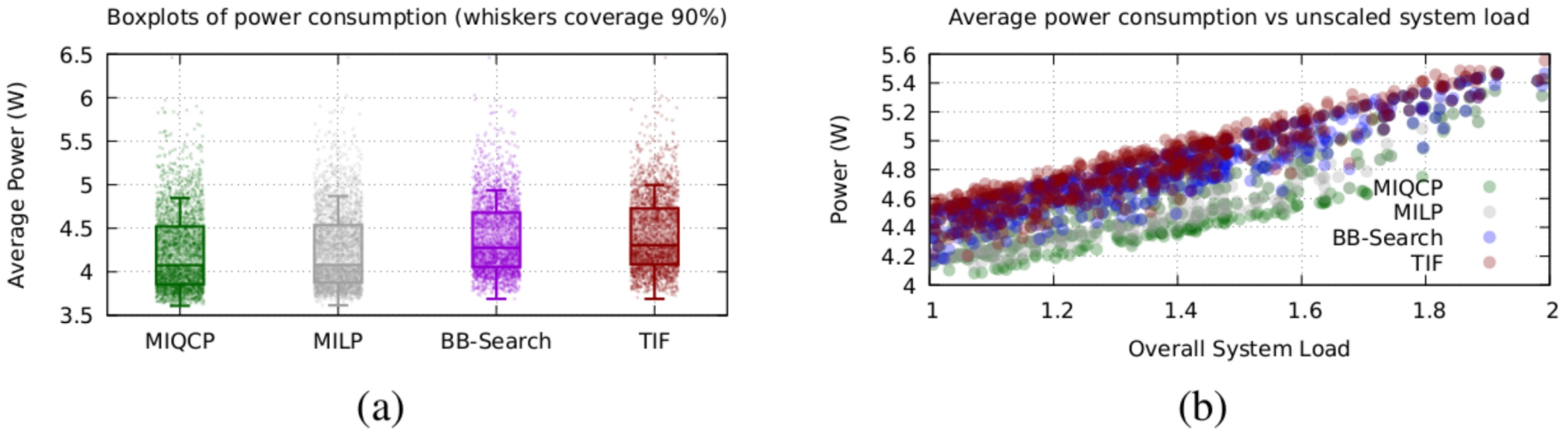


Fig. 4. (a) Boxplots of optimized average power, alongside their visual distribution, across all considered problems, under various solving strategies (different curves). (b) Detail of the optimized power as a function of the unscaled system load (X axis).

MIQCP Solver Evaluation
Power consumption optimality brings some risks

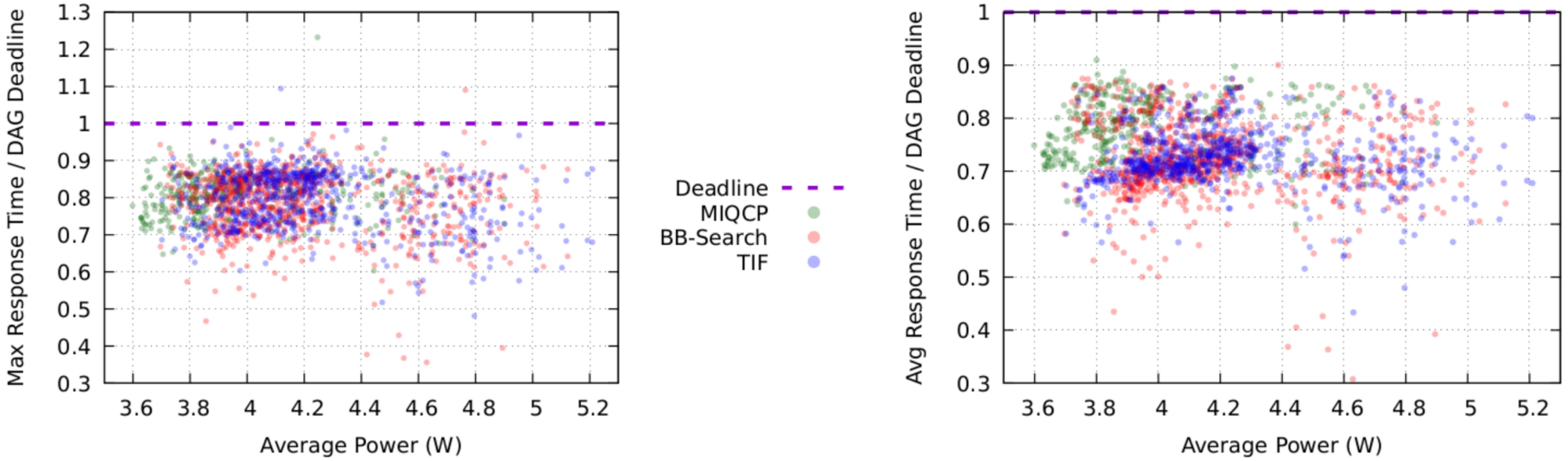


Fig. 6. Maximum (left) and average(right) relative response times (Y axis) vs average power consumption (X axis).

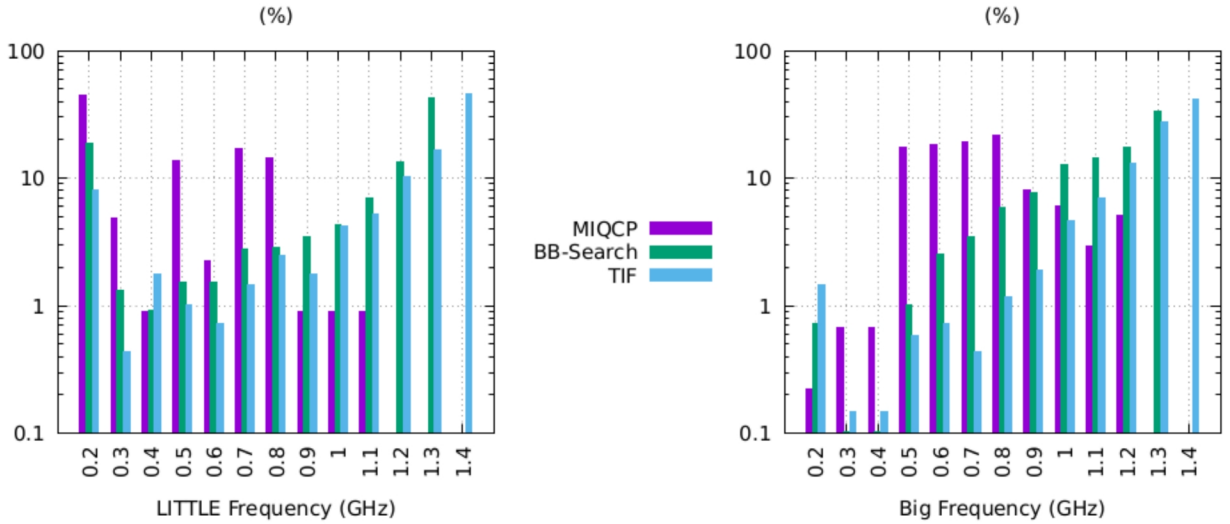


Fig. 7. Chosen frequencies for the two islands

MIQCP Solver Evaluation
Enhancing robustness of optimal config with slack maximization

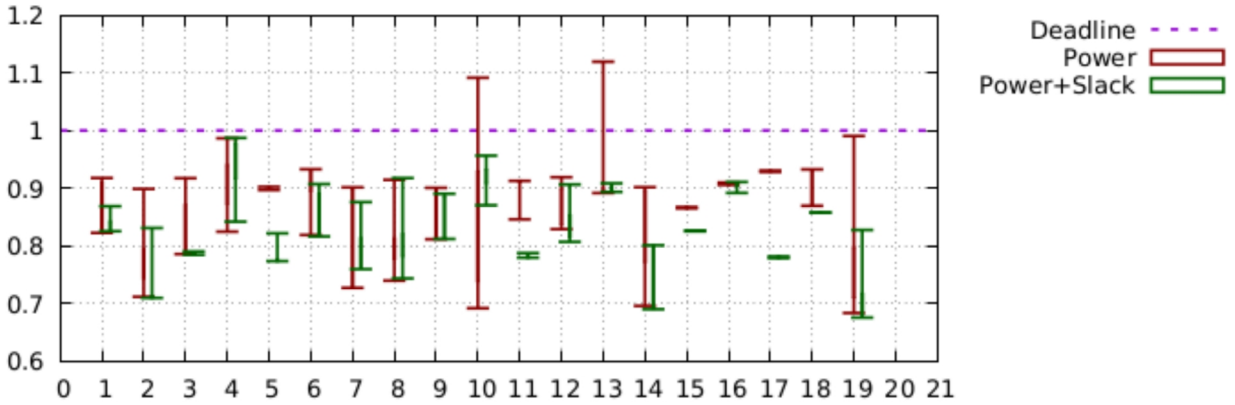
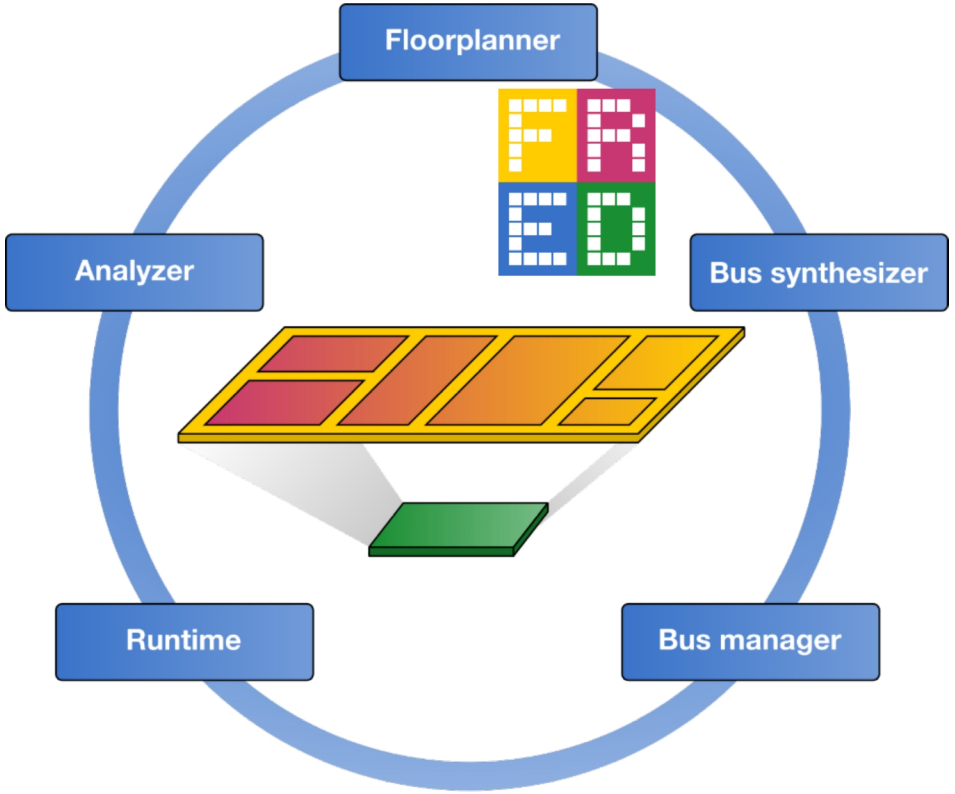


Fig. 8. Maximum response times relative to DAG periods (Y axis) for various DAGs (X axis), with and without slack optimization (green and red lines, respectively), in addition to power optimization.

FRED Framework

- Enable predictable HW acceleration on FPGA system-on-chips
- Collection of technologies developed at the ReTiS Lab

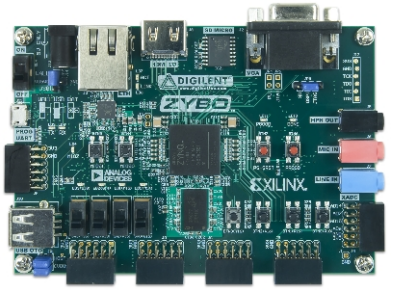
`http://fred.santannapisa.it/`

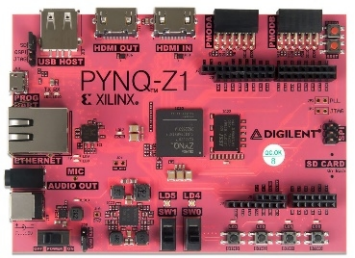


Supported platforms

Zynq Ultrascale+

Zynq-7000 series





FPGA acceleration

- Call to FPGA accelerator implemented in FRED as
  - **RPC to FRED daemon**
    - handles a queue of acceleration requests for FPGA partitions
    - allocates slots to requests
    - triggers reconfiguration of per-slot HW IPs when needed (using a recfg kernel thread to reconfigure the slot)
    - theoretical framework accounts for worst-case queueing and reconfiguration delays
- **Assigning real-time bandwidth to FRED is troublesome**
  - very small runtimes with some variability hard to predict
  - very small periods not to impact end-to-end deadlines

    → high bandwidth for FRED daemon and recfg kthread ?!?
- We just put **FRED at RT prio above SCHED_DEADLINE**, and counted for its worst-case interference as for IRQs et al. (all counted in standard inflation of WCETs already there for IRQs)

FPGA Acceleration Results

- Xilinx UltraScale+ ZCU102
- 50 randomly generated RT DAGs
- Two tasks randomly chosen to be acceleratable
  - 64x64 64-bit matmul
    - 2.12 speed-up vs SW
  - 128x128 64-bit matmul (51% of the FPGA area)
    - 7.87 speed-up vs SW
  - 22.8 reconfiguration time
- Each scenario had 300-600 activations → ~23K DAG runs
- No deadline misses in almost all scenarios
- Just one DAG was exhibiting deadline misses
  - when reoptimized for slack, it had no more misses

Selected Publications

- **Multi-Criteria Optim. of RT DAGs on Heterogeneous Platforms under P-EDF**, ACM TECS 2023
  T. Cucinotta, A. Amory, G. Ara, F. Paladino, M. Di Natale
- **Combining admission tests for heuristic partitioning of real-time tasks on ARM big.LITTLE multi-processor architectures,** Elsevier JSA 2021
  A. Mascitti, T. Cucinotta, L. Abeni
- **ReTiF: A declarative real-time scheduling framework for POSIX systems**, Elsevier JSA 2021
  G. Serra, G. Ara, P. Fara, T. Cucinotta
- **Migrating Constant Bandwidth Servers on Multi-Cores,** RTNS 2021
  T. Cucinotta, L. Abeni
- **Simulating Execution Time and Power Consumption of Real-Time Tasks on Embedded Platforms,** ACM SAC 2022
  G. Ara, T. Cucinotta, A. Mascitti
- **Energy-Efficient Low-latency Audio on Android**, Elsevier JSS 2019
  A. Balsini, T. Cucinotta, L. Abeni, J. Fernandes, P. Burk, P. Bellasi, M. Rasmussen
- **RT-Kubernetes - Containerized Real-Time Cloud Computing**, ACM SAC 2022
  S. Fiori, L. Abeni, T. Cucinotta
- **Operating System Noise in the Linux Kernel**, IEEE TC 2022
  D. B. de Oliveira, D. Casini, T. Cucinotta
- **Strong Temporal Isolation among Containers in OpenStack for NFV Services**, IEEE TCC 2021
  T. Cucinotta, L. Abeni., M. Marinoni, R. Mancini and C. Vitucci

thank you!

Q&A

t.cucinotta@santannapisa.it