Reliable but Efficient Worst Case Design for Embedded Performance MpSoC

Rolf Ernst, TU Braunschweig

with support of Robin Hapka

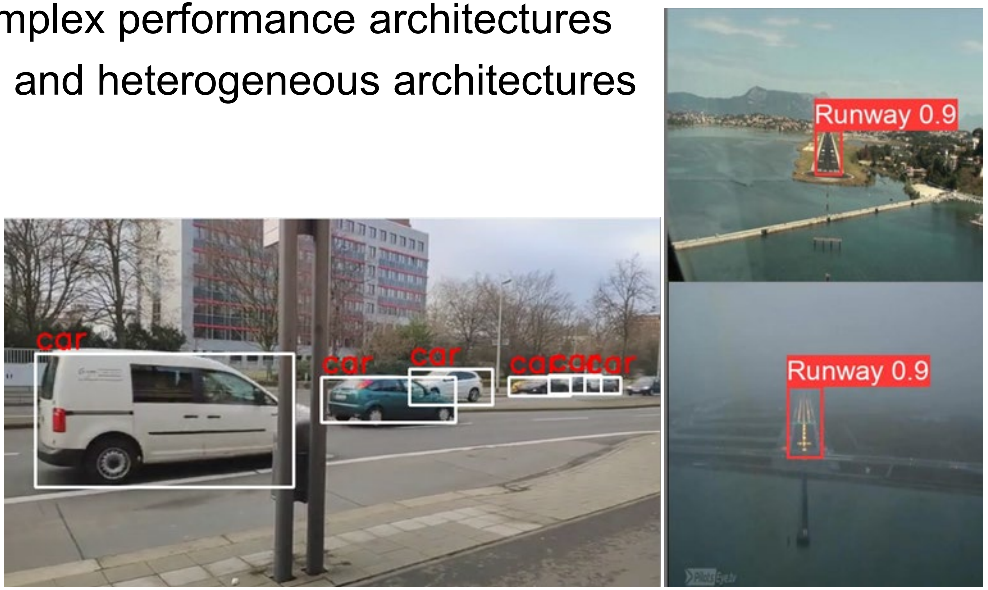
Reliable worst case design for MpSoC - background

- **performance MpSoC in critical real-time applications**
  - needed to implement high automation in vehicles, aircraft, robotics, industrial control,…
  - main application: machine learning (inference) for perception
- **implementation efficiency**
  - performance and power goals crucial to reach safety induced end-to-end deadlines (e.g. perception pipeline)

    ↔ functional safety standards require worst case design
- **last two MpSoCs**
  - experiments show limitations of formal and measurement based worst-case design (22)
  - design methods managing critical designs with timing uncertainties (23)

Efficient worst case design for embedded performance MpSoC

- **this MpSoC**
  - focused experiments for a deeper look into timing effects and modelling
  - a **new execution time model** for complex performance architectures
  - application to relevant homogeneous and heterogeneous architectures
    - **Intel Alder Lake**
    - **NVIDIA Jetson ORIN**



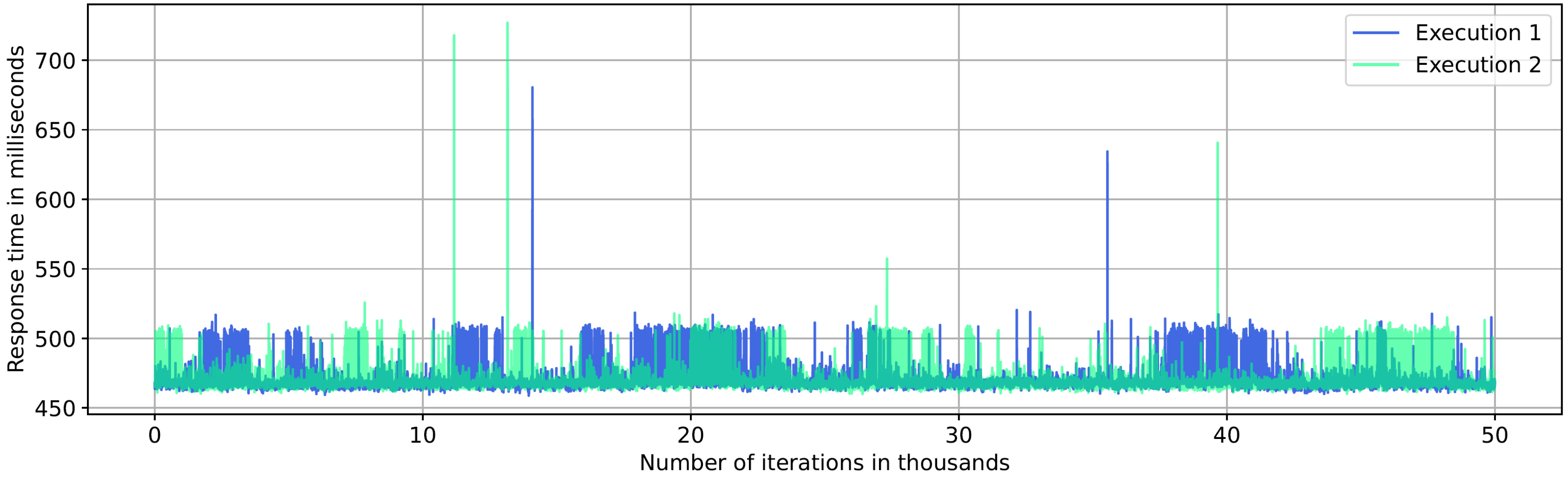
Worst case design requirements - predictability

- **timing predictability**
  - ability to predict system timing at system design time
    - formal design model, simulation, or measurements of a physical device

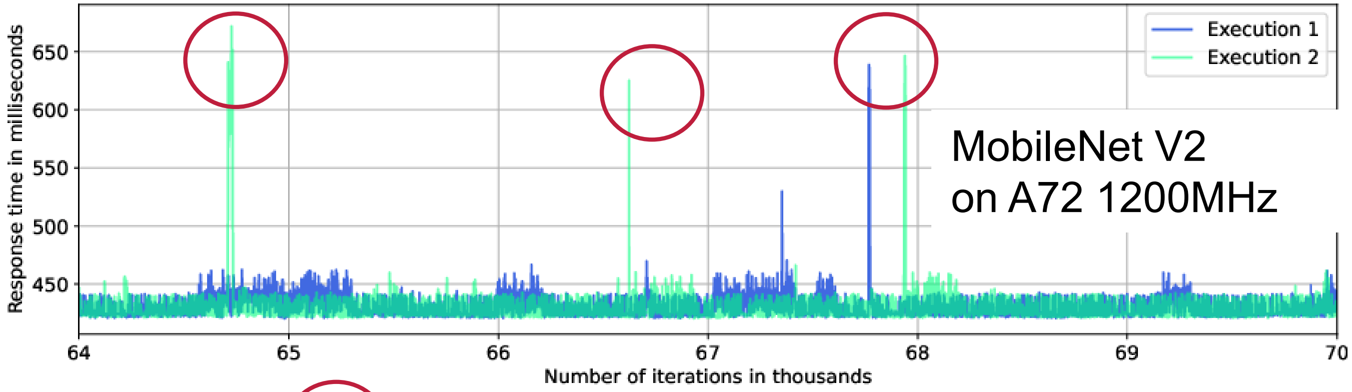
- **„traditional“ requirement for safety related real-time systems**
  - requires timing determinism for tested patterns and conditions
  - assumes repeatability – repetition of test under identical conditions leads to same result

Non-repeatable timing – MpSoC 2023

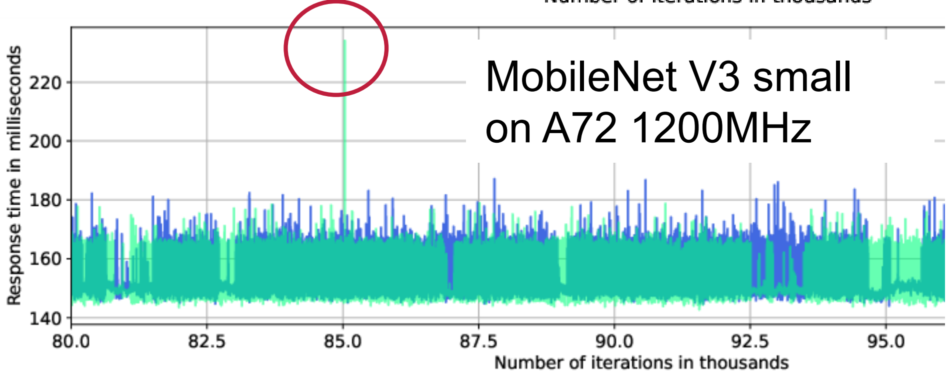
- observation: **non-repeatable response times,** despite identical execution paths
- example: A72 quadcore, 600MHz, Linux (RT, kernel version 5.10.63), Mobilenet V3 large, 2 iterative executions with 50000 algorithm iterations each



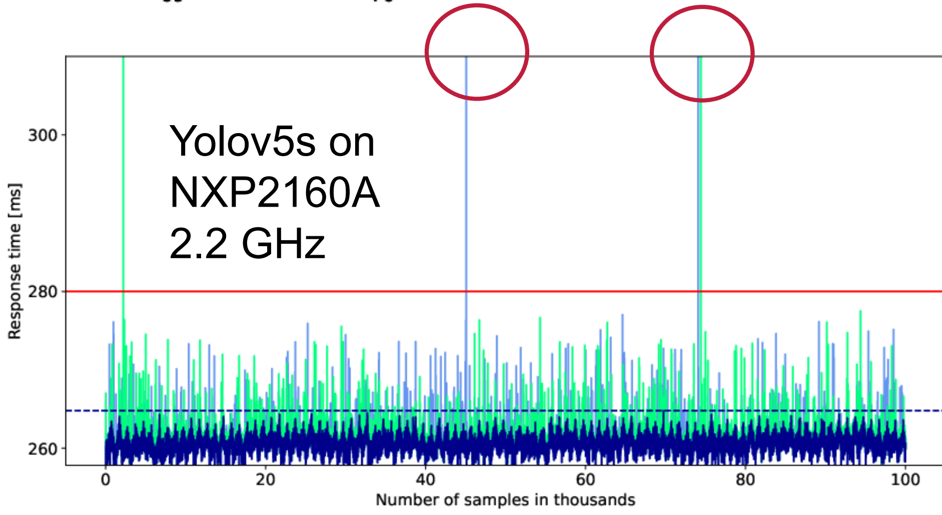
IC-individual timing – different architectures and technologies



MobileNet V2
on A72 1200MHz



MobileNet V3 small
on A72 1200MHz



Yolov5s on
NXP2160A
2.2 GHz

Worst case design - practice

- **formal deterministic timing models**
  - too many physical, architecture and software effects for sufficient accuracy
  - many unknown and/or subject to variations in production parameters and life time
- **formal probabilistic timing models**
  - mostly assume effect independence, e.g. EVT – counter examples described in literature
- **practice** mainly uses **design time measurements**
  - **good news**
    - **single timing effects** (memory hierarchy, TLB miss, …) dominating in small benchmarks are **largely masked** in tens to hundreds of million total clock cycles
  - **bad news**
    - **no execution time repeatability** of large software on complex MpSoC
    - **IC-individual timing** challenges design verification by measurement

Modeling response times

- **no straightforward deterministic or probabilistic timing**

- **possible interpretation: execution time variation is combination of two sources**

1. **large number of superimposed individual events** (arbitration, cache protocols, stateful timing (DRAM), virtual addressing, ..) with small individual impact per event (<100k clock cycles)
   - such events were focus of traditional WCET investigations

2. **very few large effects** causing **rare outliers** with millions of cycles

Try novel execution time model and design approach

- **category 1** should approach ***normal distribution***, if superimposed events are independent
  - use **Gaussian execution time model –** execution time variation appears as ***"noise"***
  - physical HW effects will cause **randomness**
  - **include minor outliers** which may be caused by dependencies of category 1 effects

- **category 2** of remaining rare outliers could be treated in a number of ways:
  - **include** in WCET & trust in **extreme value statistics** (EVT) for **exceedance probability**
  - **change design or configuration** to reduce/suppress effects
  - **design for timing fault tolerance**, e.g. with redundancy mechanisms -> **timing diversity (MpSoC 2023)**

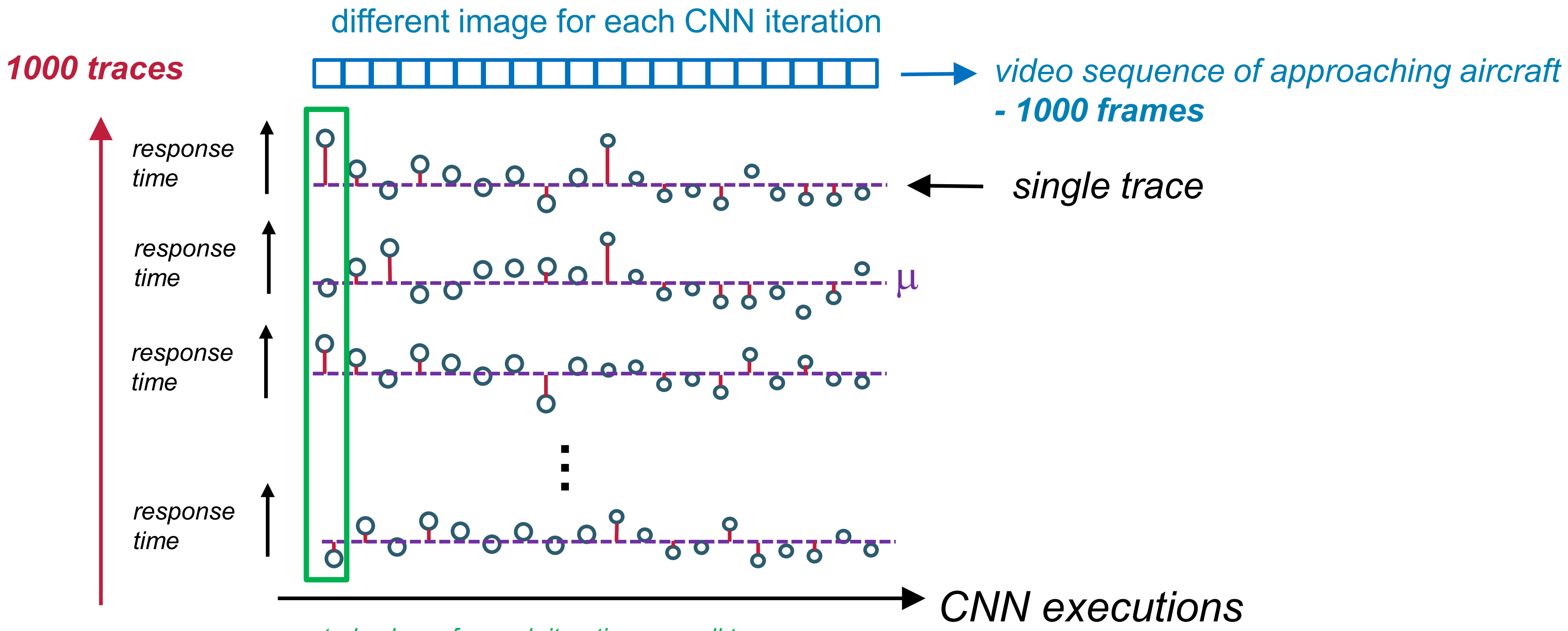
Experiments

- **homogeneous: Intel NuC i5-1240p**
  - 2x 4 P-Cores, 8 E-Cores (Alder Lake)
  - 2 levels private cache + shared LL cache
  - Intel 7 technology

- **heterogeneous: NVIDIA Jetson Orin AGX**
  - focus automotive
  - 12x ARM Cortex-A78AE, GPU 64 tensor nodes, 2xDLA (not used)
  - Samsung 8nm technology

- **application YOLOv5**
  - avionic runway perception video trace

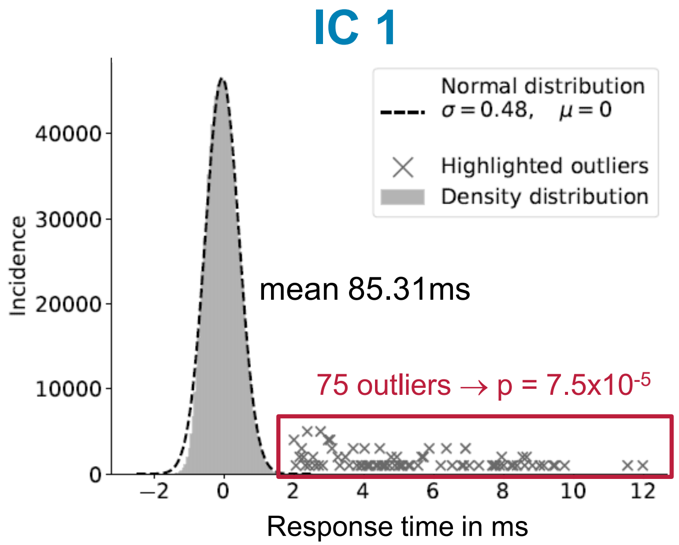
Test setup

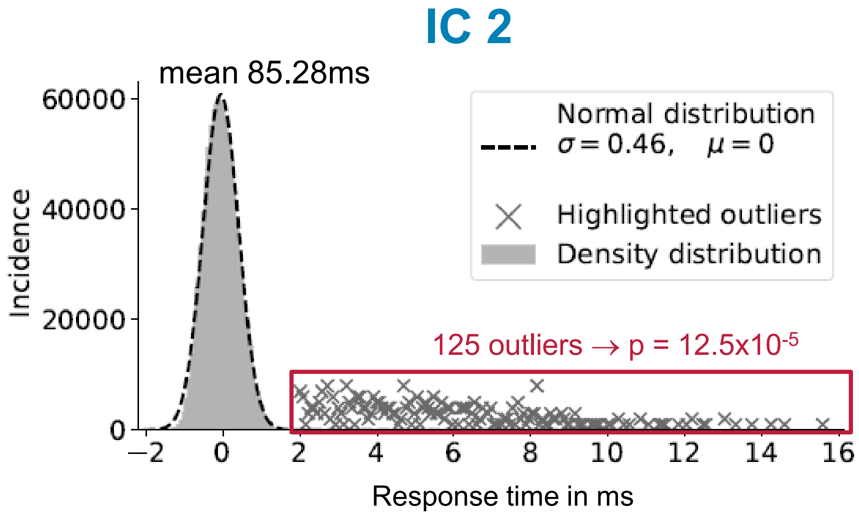


expected value μ for each iteration over all traces

- *covers stateful timing behavior and data dependencies*

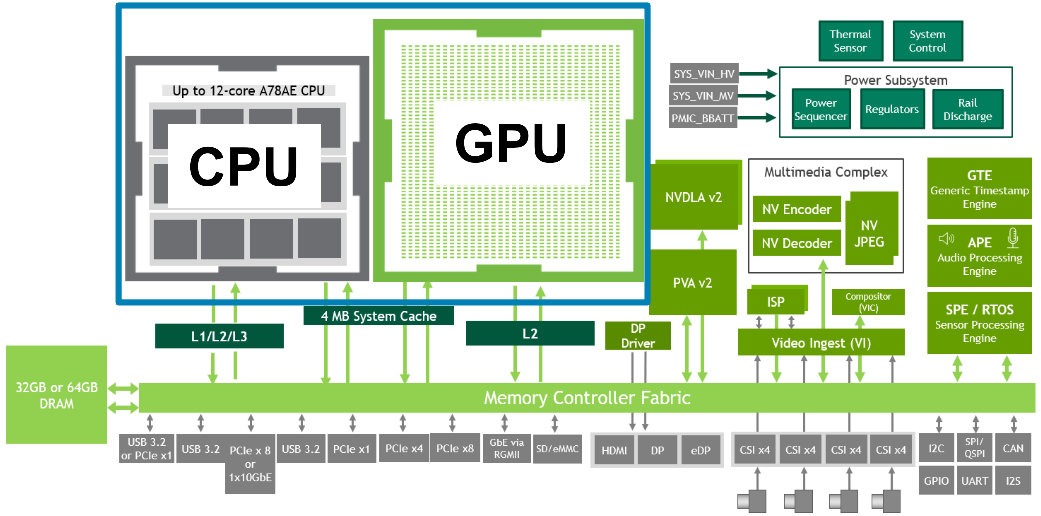
Experiment 1 – Intel NuC i5-1240p – two different ICs



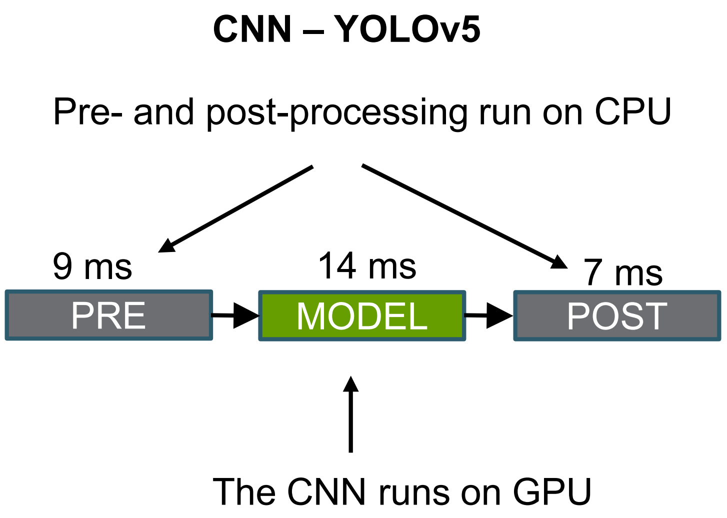


- 1000 traces with 1000 CNN executions per experiment
- all variations above 2ms counted as outlier
- additional tests autocorrelation, power spectral density, … in [1]

Experiment 2: nvidia ORIN AGX

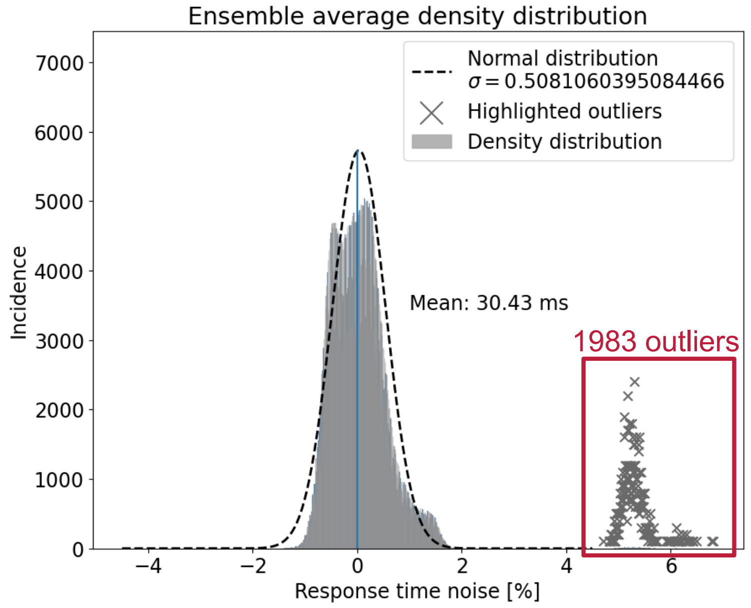


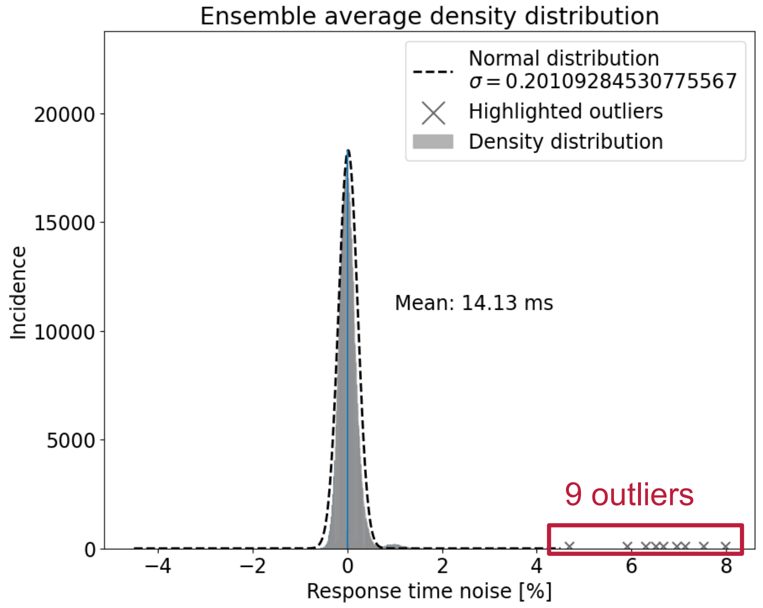
Source: https://developer.nvidia.com/blog/delivering-server-class-performance-at-the-edge-with-nvidia-jetson-orin/



Sequential execution – end-to-end

skip first frame subject to rail gating (controllable „large“ effect)



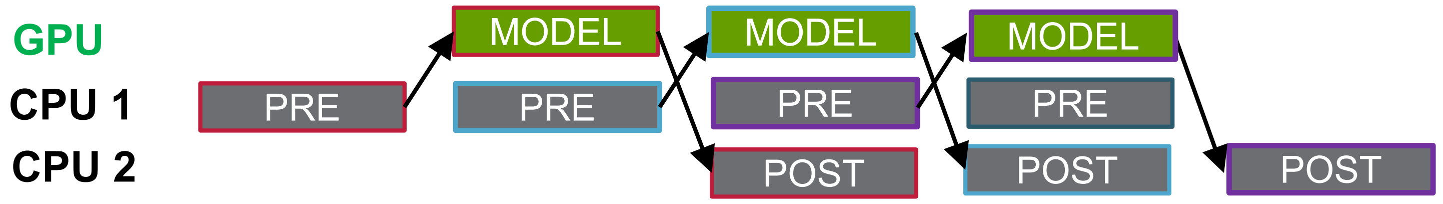


Low Power: 15W

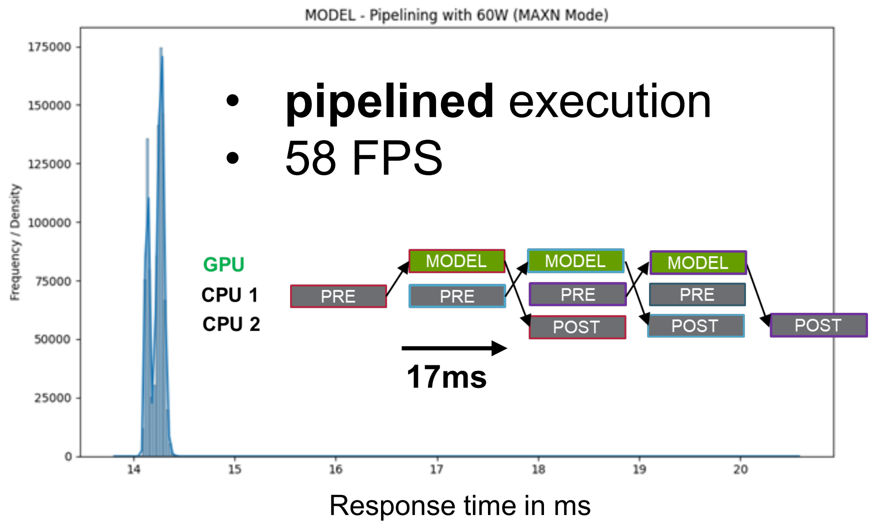
Max Power: 60W

Nvidia hardware – tune with pipelining

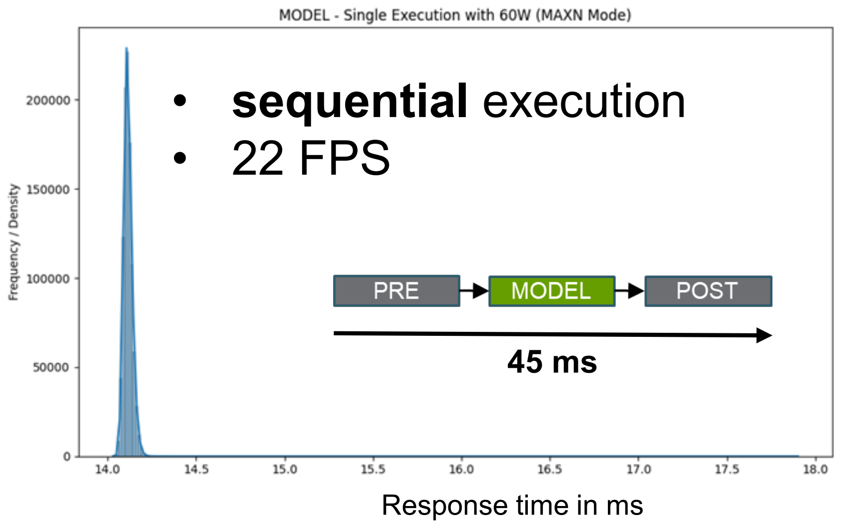
- **overlap PRE – MODEL and POST**
- **leads to resource sharing effects**



Comparison sequential ↔ pipeline



- **pipelined** execution
- 58 FPS



- **sequential** execution
- 22 FPS

- max performance (MAXN mode) 60W
- 2 narrowly superpositioned normal distributions – likely due to job release jitter
- **memory interference did not invalidate modeling w normal distributions**

Category 2 - Outliers

- **major outliers rare and uncorrelated in all experiments**
  - justifies separate handling – depends on application and strategy (see above)
  - design can mask outliers with redundant channels → Timing Diversity [2, 3]

- **exception: „invasive“ chip control, such as IC temperature management with DVFS**
  - persistent timing effects [1]
  - must be **deactivated** or must be treated as **safety case**

Conclusion

- **response time of CNNs on performance MpSoC closely follows normal distribution**
  - for homogeneous and heterogeneous architectures
  - major resource interference shifts mean value
  - rare outliers uncorrelated – to be handled in application or architecture

- **basis of a novel probabilistic execution time model**
  - enables efficient utilization of MpSoC performance at high confidence

- **can be used for design and monitoring**

Related papers

[1] R. Hapka, R. Ernst, “Conservative Design with High-Performance COTS Architectures - Beyond Traditional Approaches”. To appear at IEEE Conf. on Emerging Technologies and Factory Automation (ETFA), Sep. 2024, Padua

[2] Robin Hapka, Anika Christmann, Rolf Ernst, Alexander Kuzolap, Peter Hecker, Marius Rockschies, Martin Halle, Frank Thielecke. “Safe Usage of Multi-Cores in Neural-Network Avionics Applications”, IEEE Digital Avionics Systems Conference (DASC) 2023, October 2023, Barcelona.

[3] Anika Christmann, Robin Hapka, Rolf Ernst, “Formal Analysis of Timing Diversity for Autonomous Systems”, IEEE DATE 2023, April 2023, Antwerp, Belgium.