# Identification of dual processes using side-channels

Niklas Lindskog, Ericsson Research, Lund, Sweden

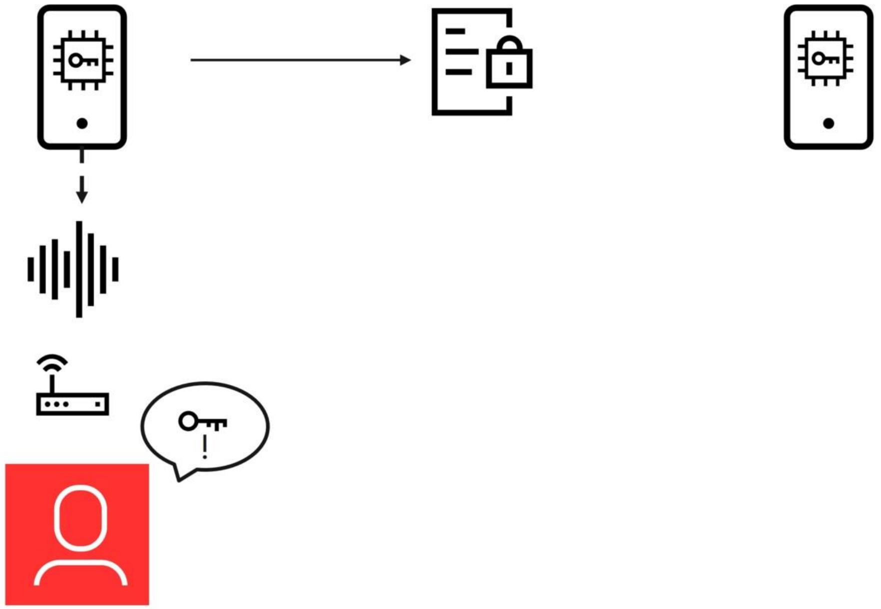
Jakob Sternby, Ericsson Research, Lund, Sweden

Håkan Englund, Ericsson Research, Lund, Sweden

# Side-channels – in a nutshell

- A side-channel:
  - Is a covert channel to extract information about the internal state of electronic device.
  - Exists because there is a correlation between the internal state of the device and an observable information leakage.
  - Leakage be component-specific or target an entire device.
  - Leakage be utilized by an attacker to extract secret data, e.g., cryptographic keys
  - Leakage can also be utilized to monitor the health of a device or a software process.

- Exists in many different variants
  - Logical (Memory access patterns, network traffic, etc.)
  - Physical (Power consumption, electromagnetic emissions, heat, etc.)

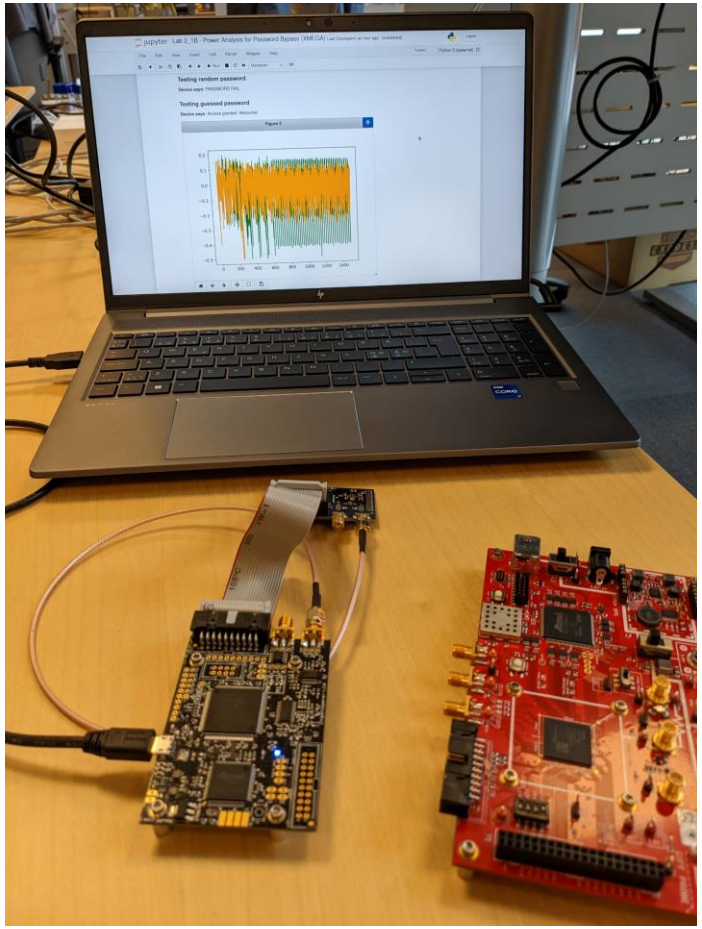


# Extracting physical side-channels

- Power consumption, electromagnetic emissions, heat, acoustics etc.
- Can be measured locally by external probe or by telemetries in device.
- Power consumption
    - Measured using a resistor shunt in the VCC line.
    - Can determine fluctuations in power consumption.
    - Granularity can be weakened by capacitor.
    - Can also be measured from the “inside” by power management unit.
- Electromagnetic emissions
    - Measured using an electromagnetic probe.
    - Strength varies with power consumption.
    - Noisier but can focus on part of chip.
- Heat
    - Thermal camera
    - Can also be measured inside device.

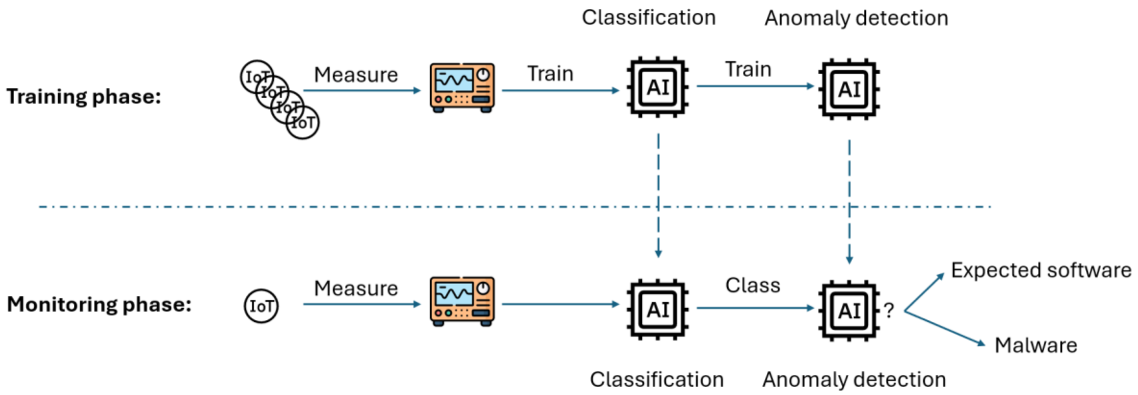
# Side-channel monitoring

- Leverage covert channel to protect the device
  - Enables runtime monitoring of processes without presence on device.
  - Possible to retro-fit on existing equipment.
  - Very difficult for attackers to stay undetected in side-channel emissions.
- Almost same procedure as side-channel analysis
  - Less focused on data, more on instructions
  - Goal is to find anomalies compared to expected behavior
- Preferable to avoid repetitions – one-shot analysis
  - Most attacks on cryptographic keys rely on repetition
- Can be used both as standalone solution and as complement to conventional security solutions.



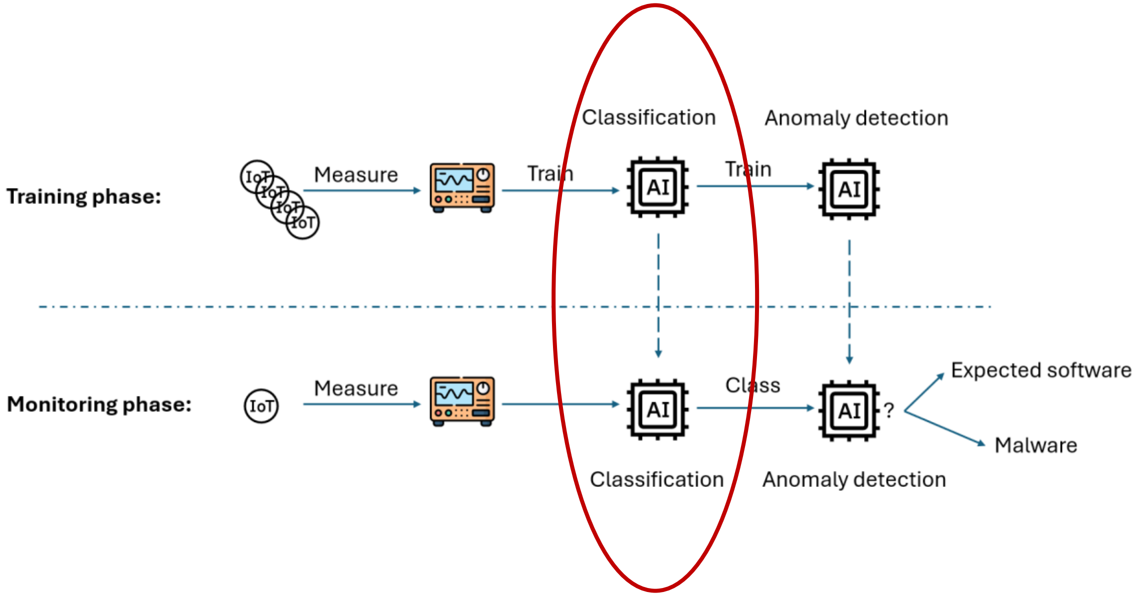
# Multi-process side-channel monitoring

- What if expected behavior of device is many different simultaneous behavior?
- Process becomes twofold
  - Determine which process that is executing
  - Is the determined process executing correctly?
- Anomaly detection in presence of uncorrelated processes is studied in the existing literature
  - Classifying multi-processes is not well-studied



# Multi-process side-channel monitoring

- How do we separate a side-channel measurement into multiple processes?
- Alternatives:
  - Multiple probes
    - Possible but increases complexity of monitoring
  - Create super-classes (i.e., A + B, A + C, etc.)
    - Only possible if processes are correlated
  - Learn to separate two processes from single measurement
    - Encode processor characteristics as a part of classification?
    - Our approach



# Research questions

- We want to determine
  - Is the possible to perform multi-process anomaly detection using side-channels?
    - Is it possible to monitor two distinct software processes executing simultaneously using a single side-channel probe?
    - Assuming we can classify the processes, can we determine which process is running on which processor?

# Identification of dual processes using side-channels

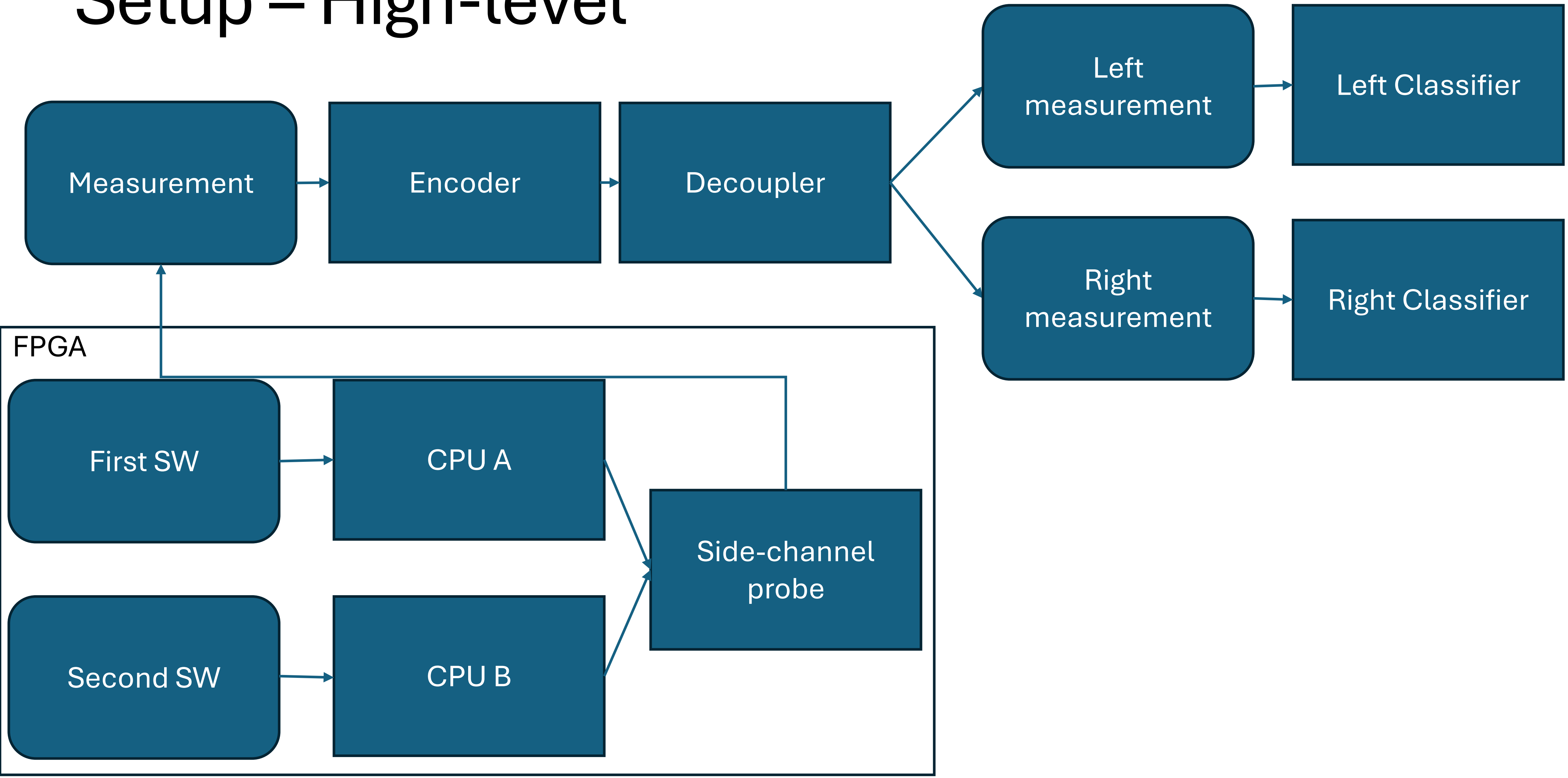
# Our contributions

- Our contribution is three-fold:
  - Improving the practicability of multi-process side-channel monitoring
    - Multi-model machine learning solution
  - Evaluation of feasibility in performing side-channel monitoring on multiple, simultaneously executing, software processes.
  - Machine learning-based approach to classify a single side-channel measurement trace as two classes from a set of predefined processes.
    - Wherein the two classes also indicates which of the two cores the process has been executed on.

# On a high-level

- Execute simultaneous processes on separate cores
- Power consumption of device measured using single probe
- Use encoder to create latent representation
- Create concept of “left” and “right” measurement
- Classify respective side
- Give classification as input to anomaly detector [out of scope]

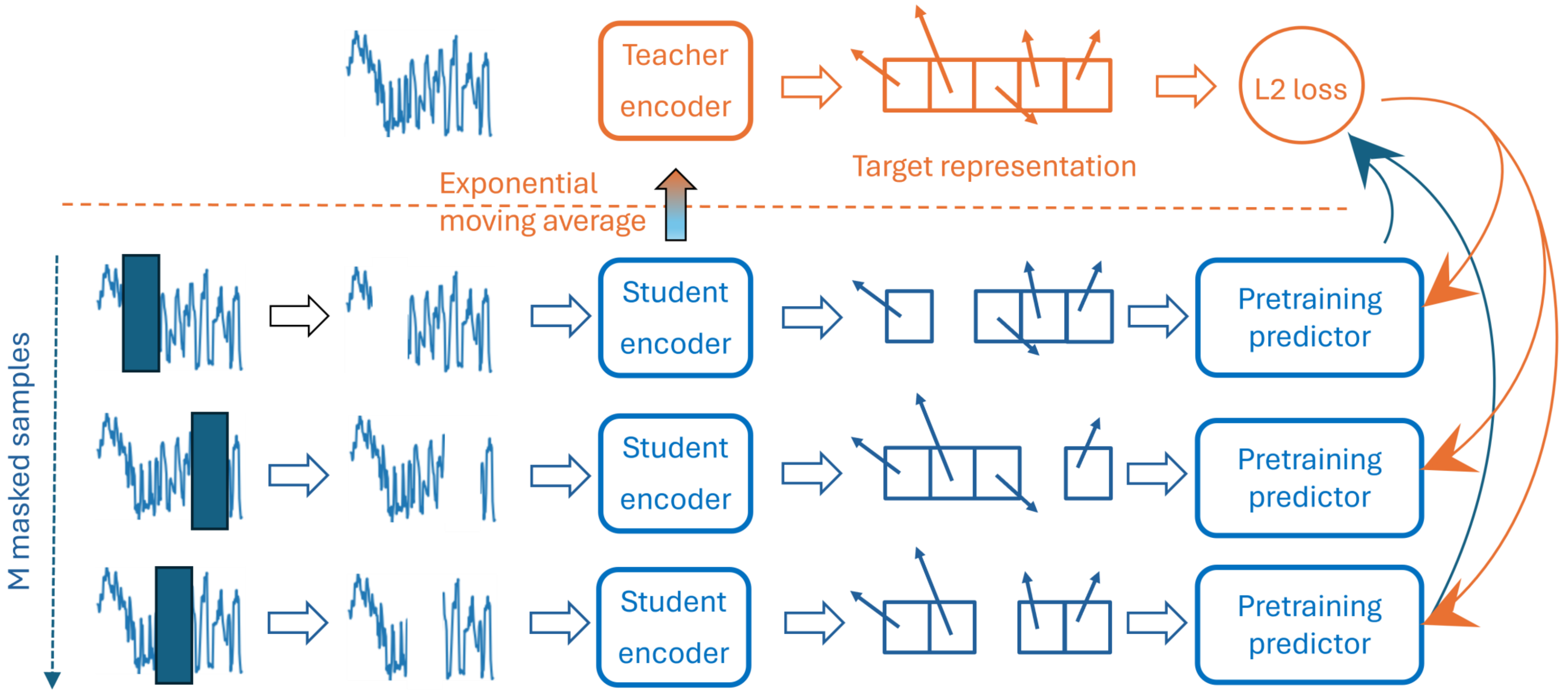
# Setup – High-level



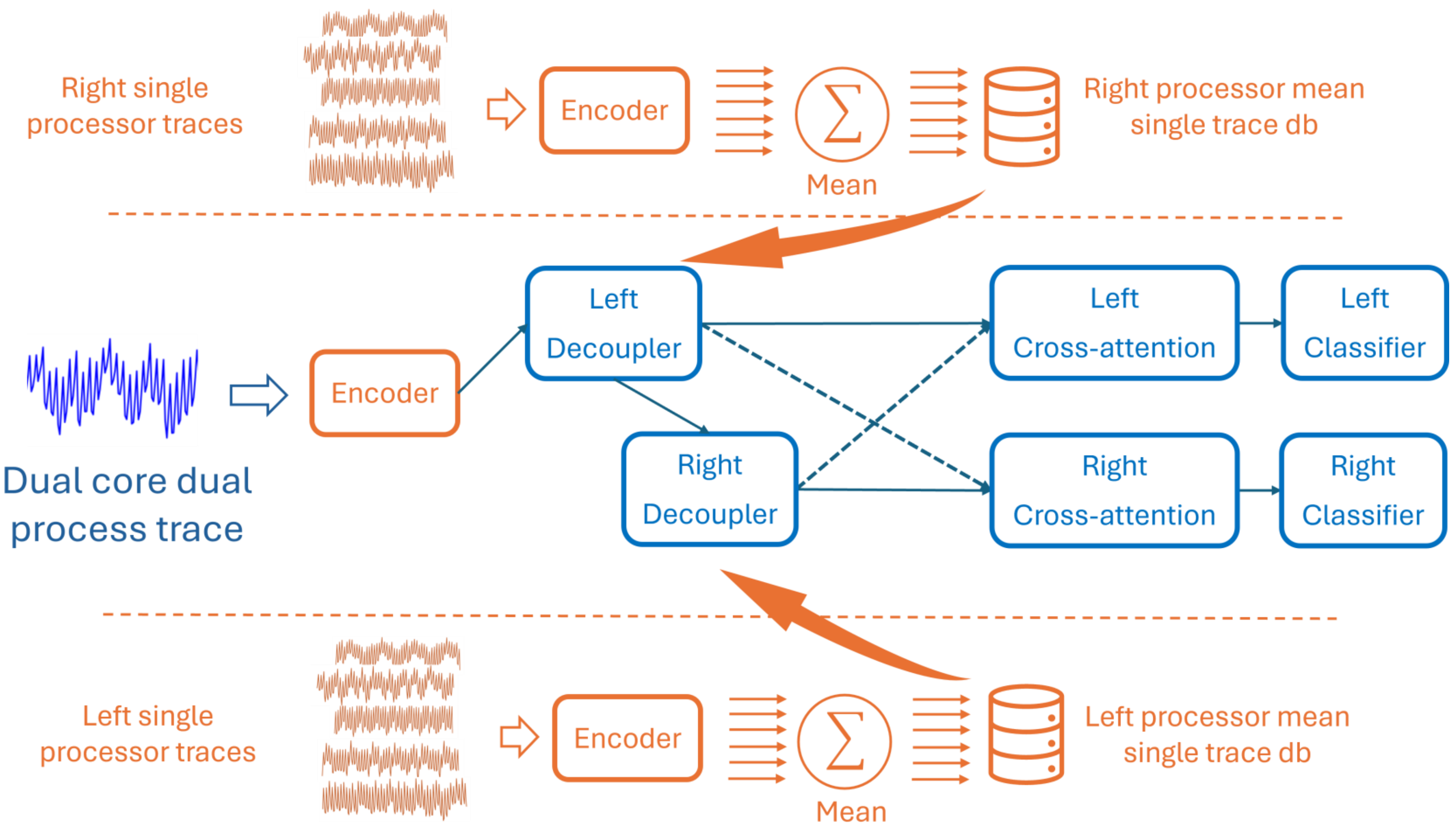
# Setup – Machine learning models

- Four blocks:
  - Pretrained encoder
    - Separately trained, produce well-suited latent representation of side-channel-trace
  - Decoupler
    - Divide the encoded trace into a left and right block
  - Cross-attention
    - Mix attention of the left and right block
  - Classifier
    - Determine and output two classes, one for left and one for right processor.

# Setup – Pre-training encoder



# Setup – Machine learning models



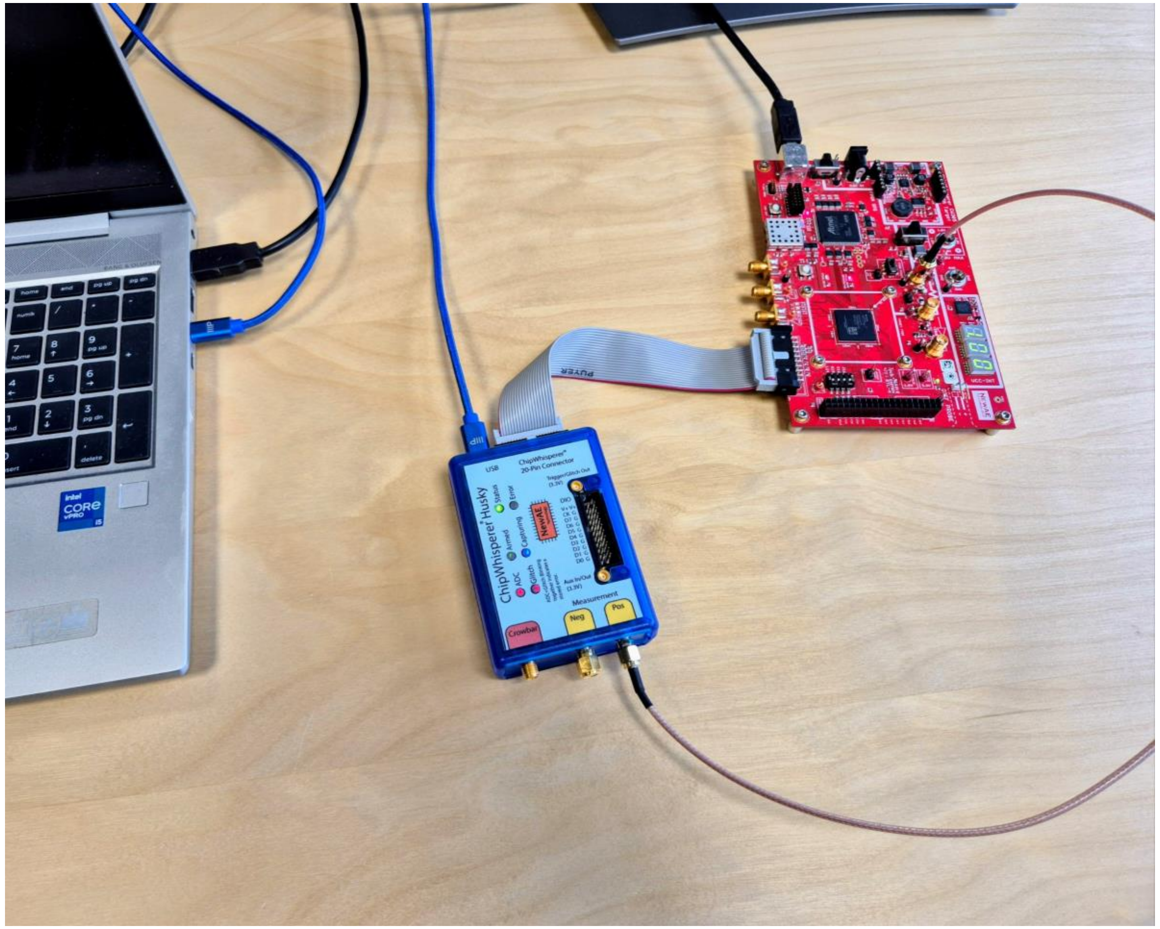
# Setup – Attention heads

- Decoupler has one attention head per known process (class)
  - Supervised
  - Learns specific traits for process and class
- Cross-attention has additional attention heads
  - Unsupervised
  - Learns specific dual-core properties

# Evaluation & Results

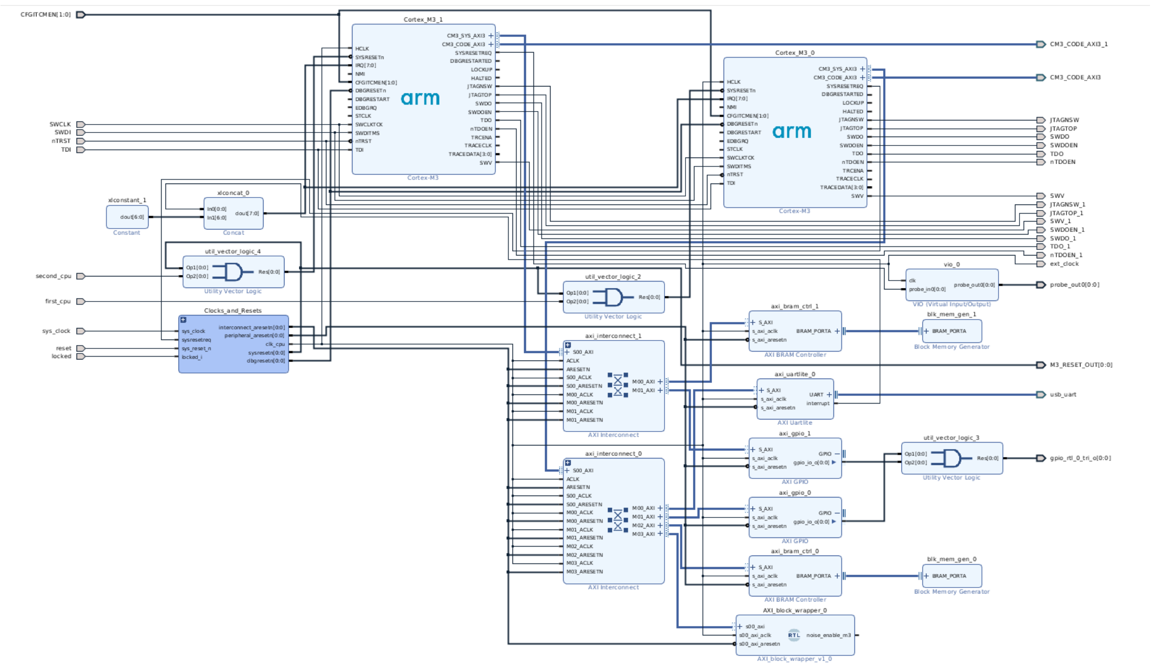
# Measurement setup

- Monitor
  - Chipwhisperer Husky + HP Elitebook 850 G8
- Device-under-monitoring
  - CW305 A100 – Artix-7 FPGA
  - Pre-soldered resistive shunt on VCC
- Synchronous monitoring
  - 4 samples / clock cycle



# Measurement setup (cont.)

- Two soft Cortex-M3 cores, $C_A$ and $C_B$
  - Clock speed of 20 MHz
  - Separate BRAM blocks
  - Clock generated by monitor
- Measurement trigger
  - Wire triggered from $C_A$



# Scope limitations

- Simple software
  - Evaluated using software from BEEBS*
    - Software without branches, interrupts or varying input
- Process synchronization
  - Simultaneous start of processes
- “Fresh” processor
  - Processors in reset prior to measurement
- Deterministic processors
  - No branch-prediction or speculative execution

* Bristol / Embecosm Embedded Benchmark Suite (https://github.com/mageec/beebs)

# Evaluation

- Eleven classes
  - Ten software processes + one category of “no process”
- ~ 1000 samples of 4500 clock cycles per class combination
- 70% used for training
- 10% for validation
- 20% as held-out independent test set

# Results

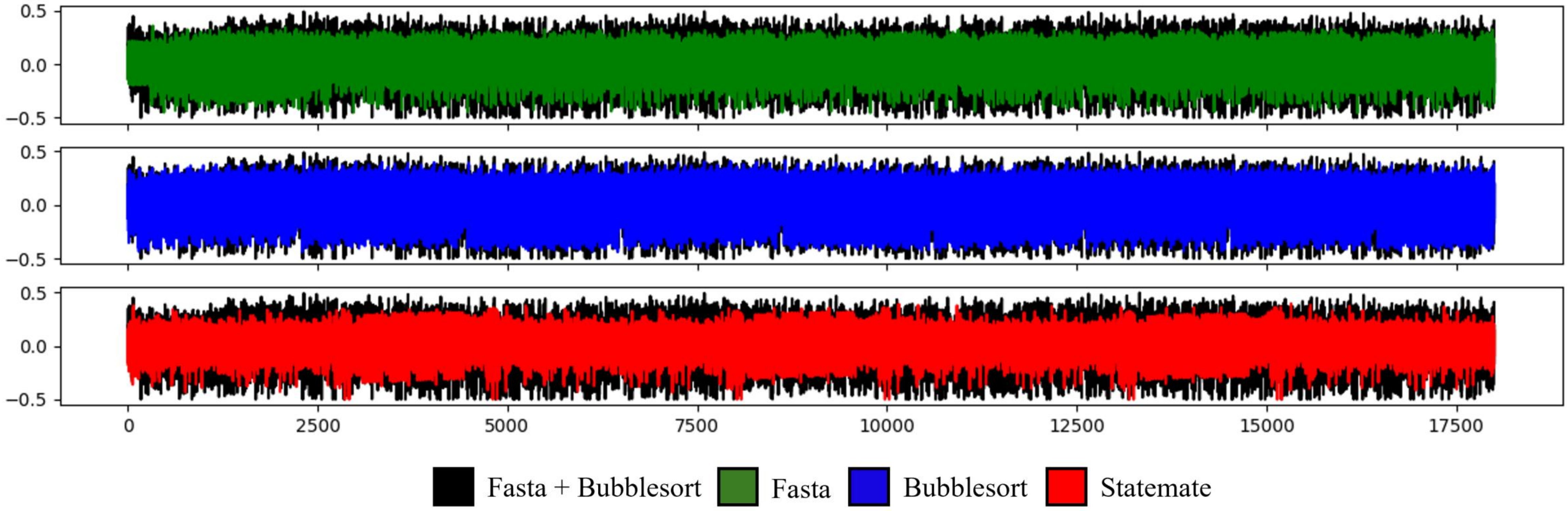
- Five-fold validation
- Average dual correctness – 94,1%
  - Worst process – 85,4%
- Average single correctness – 97%
  - Worst process – 92,7%

TABLE I. DUAL-CORE CLASSIFICATION ACCURACY LISTED BY THE CLASS OF THE PROCESS RUNNING ON $\mathcal{C}_{\mathcal{A}}$.

| $\mathcal{C}_{\mathcal{A}}$ class | Accuracy | Single acc. | # Test | # Valid | # Train |
|---|---|---|---|---|---|
| cnt | $96.1\% \pm 0.9$ | $98.0\% \pm 0.5$ | 2200 | 1100 | 7700 |
| fasta | $91.1\% \pm 1.1$ | $95.6\% \pm 0.5$ | 2000 | 1000 | 7000 |
| prime | $97.5\% \pm 0.4$ | $98.8\% \pm 0.2$ | 2200 | 1100 | 7700 |
| ahacompress | $95.8\% \pm 4.8$ | $97.9\% \pm 2.4$ | 2200 | 1100 | 7700 |
| bubblesort | $97.3\% \pm 4.2$ | $98.6\% \pm 2.1$ | 2200 | 1100 | 7700 |
| cover | $98.2\% \pm 4.1$ | $99.1\% \pm 2.0$ | 2200 | 1100 | 7700 |
| tarai | $91.9\% \pm 2.0$ | $95.9\% \pm 1.0$ | 2200 | 1100 | 7700 |
| lcdnum | $96.3\% \pm 3.6$ | $98.2\% \pm 1.8$ | 2200 | 1100 | 7700 |
| crc32 | $85.4\% \pm 3.6$ | $92.7\% \pm 1.8$ | 2200 | 1100 | 7700 |
| statemate | $96.4\% \pm 5.0$ | $98.2\% \pm 2.5$ | 2200 | 1100 | 7700 |
| idle[a] | $88.5\% \pm 5.0$ | $94.2\% \pm 2.5$ | 2200 | 1100 | 7700 |
| Total | $94.1\% \pm 2.2$ | $97.0\% \pm 1.1$ | 24000 | 12000 | 84000 |

[a]No process currently executing on $\mathcal{C}_{\mathcal{A}}$.

# Measurements illustrated



# Discussion & Next steps

# Discussion

- Results indicate that dual processes monitoring is feasible
  - Latent representation is key.
  - Can we represent more complex software behavior?
- Oversampling
  - Often not feasible in real-world settings.
  - How would it impact result if we go below Nyquist threshold?
- Processor optimizations
  - Not only software but also processor behavior must be monitored.
  - Can we learn to identify hardware optimization patterns?

# Next steps and interesting research directions

- End-to-end anomaly detection
  - Select golden sample from classifier output
- More than two processors
  - What is the noise limit?
- Single process on multiple cores
- What can we represent in latent space?
  - We must evaluate whether we efficiently can determine (or filter out)
    - Hardware optimizations
      - Out-of-order execution
    - Software branches
      - Combine with prior work on side-channels & control flow graphs

# Thank you for listening!
# Questions?

ERICSSON

ERICSSON

https://www.ericsson.com/en/blog/2023/4/side-channel-analysis