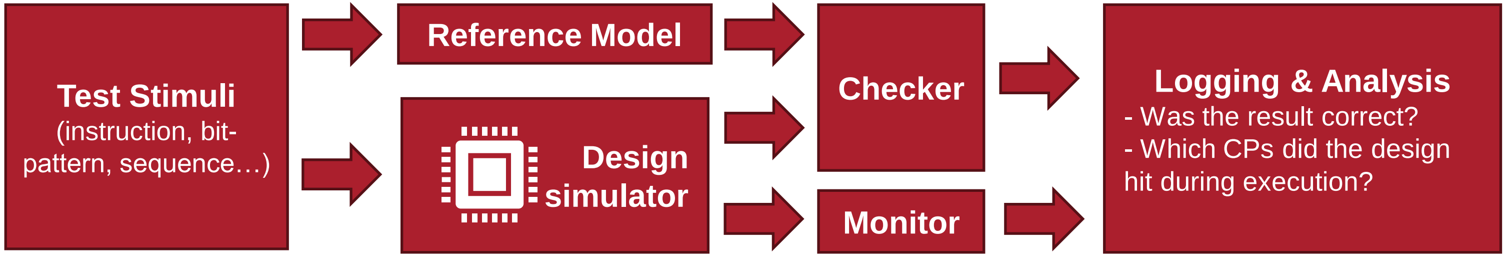
A SURVEY OF RESEARCH ON MACHINE LEARNING FOR TEST-BASED VERIFICATION

Dr Chris Bennett

9th April 2025

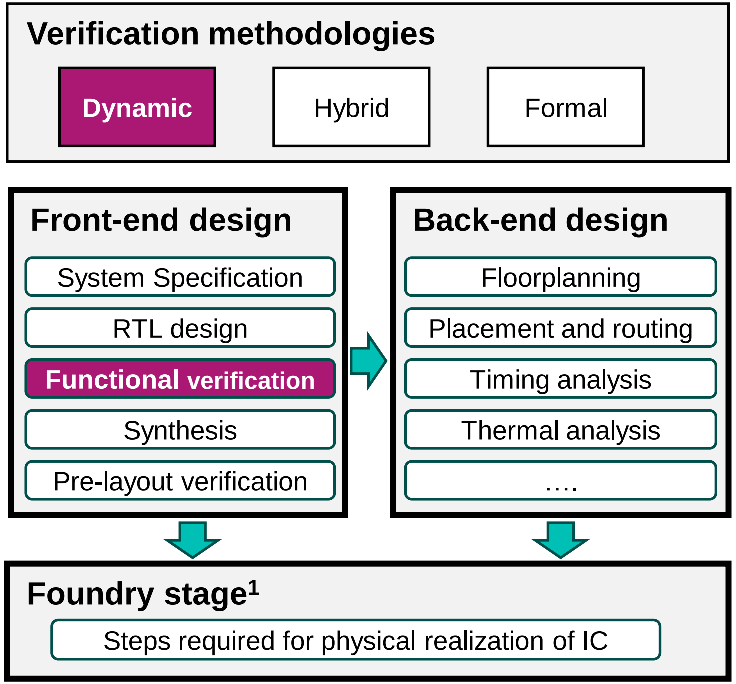
Background

- **Test-Based Functional Verification** is a process to establish a **design** carries out functions according to its **specification**.
- **Coverage points (CPs)** analogous to named **states** of a **design**
- **Challenge** is to choose the inputs to hit all specified coverage points using minimal resources ⟶ use machine learning



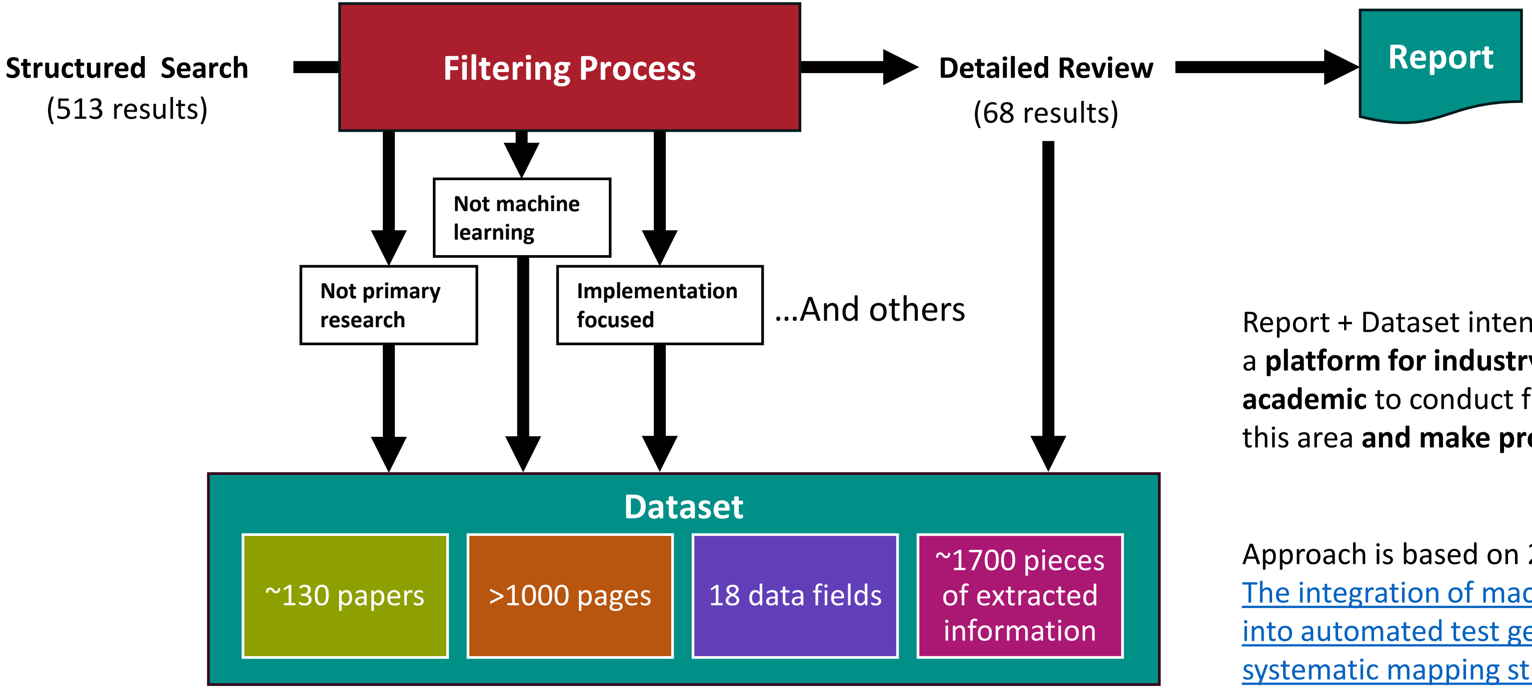
Part of a bigger picture

- **Functional verification** using ML-based **dynamic** (test) methods for micro-electronic **designs**
- Research increasing…but most ML test-techniques not adopted
  - What actions are needed to make **better use of existing** techniques
  - Lessons for research into **new techniques**



1 Danciu, G. M., & Dinu, A. (2022). Coverage Fulfillment Automation in Hardware Functional Verification Using Genetic Algorithms. *Applied Sciences*

Sourcing the data



Report + Dataset intended to provide a **platform for industry and academic** to conduct future work in this area **and make progress**

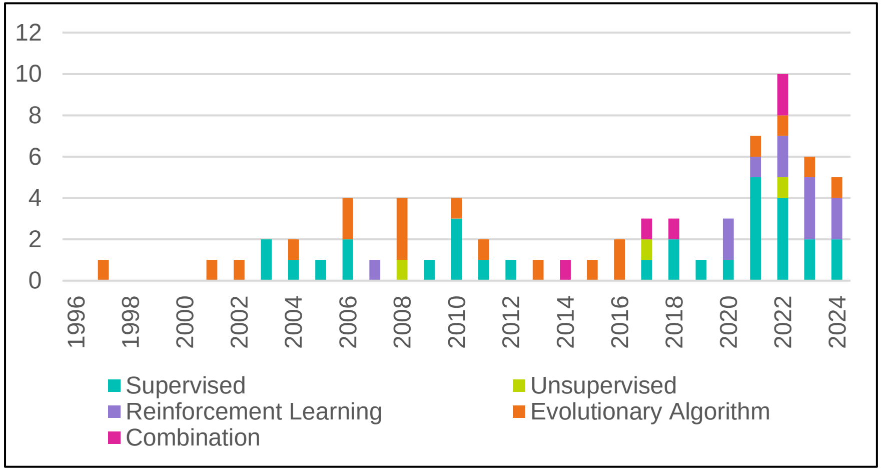
Approach is based on 2023 paper: [The integration of machine learning into automated test generation: A systematic mapping study][1]

[1]Fontes, A., & Gay, G. (2023). The integration of machine learning into automated test generation: A systematic mapping study. *Software Testing, Verification and Reliability*

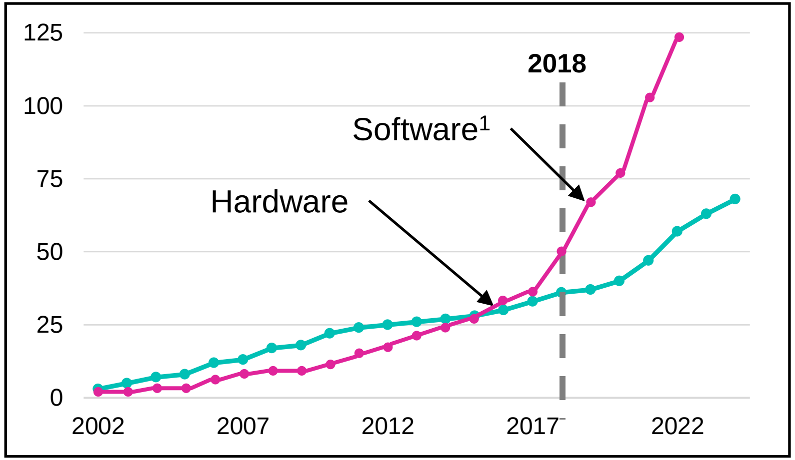
Historical trends

- In research for 25 years+
- We include Evolutionary Algorithms

- Interest in techniques changes
- Use of ML lags other areas?



Techniques by year and type



Cumulative sum of research using ML for test-based verification

[1]Fontes, A., & Gay, G. (2023). The integration of machine learning into automated test generation: A systematic mapping study. *Software Testing, Verification and Reliability*

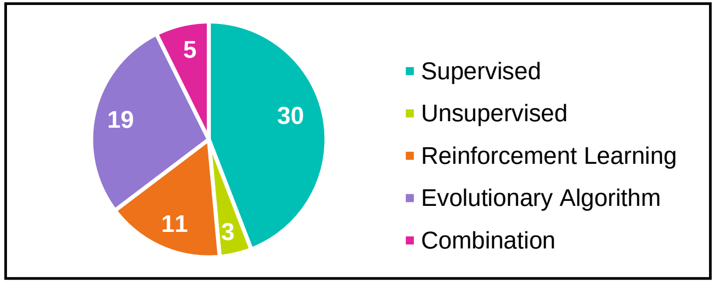
Lots of research but low adoption

- **Reason one:** using ML to map coverage to test stimuli is difficult:
  - Lack of
    - positive training examples
    - distance metrics in coverage space and test space
    - enough positive training examples
  - Stimuli and micro-architecture behaviour at different abstraction levels
  - Design changes alter the ground truth relationships

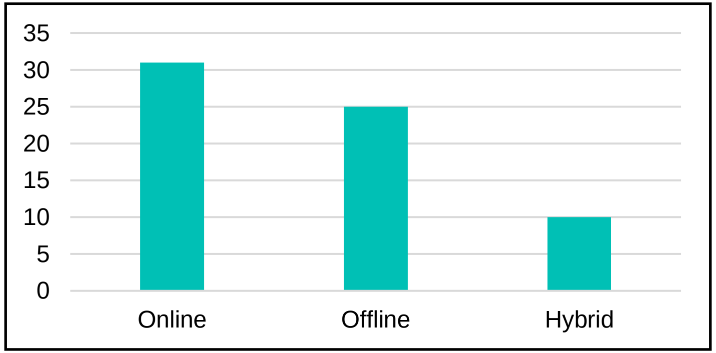
A variety of techniques used

- Algorithm types
  - **Supervised:** input-output pairs
  - **Unsupervised:** patterns in data
  - **Reinforcement:** trial and feedback
  - **Evolutionary:** optimising sets of inputs

- Training methodologies
  - **Online:** update model after every simulation
  - **Offline:** train once with historical results
  - **Hybrid:** bootstrap with historical results and update model regularly



Count of research by algorithm type



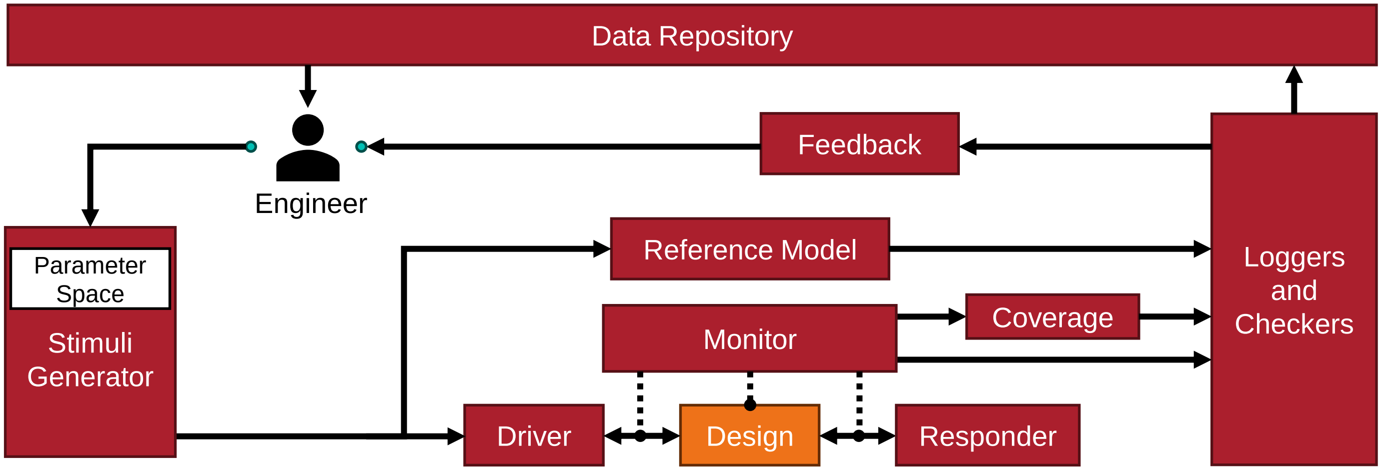
Count of research by training method

Lots of research but low adoption

- **Reason one:** using ML to map coverage to test stimuli is difficult:
  - Lack of
    - positive training examples
    - distance metrics in coverage space and test space
    - enough positive training examples
  - Stimuli and micro-architecture behaviour at different abstraction levels
  - Design changes alter the ground truth relationships
- **Reason two:** too much choice has created a confusing landscape for practitioners ⟶ classify research to aid adoption of ML techniques

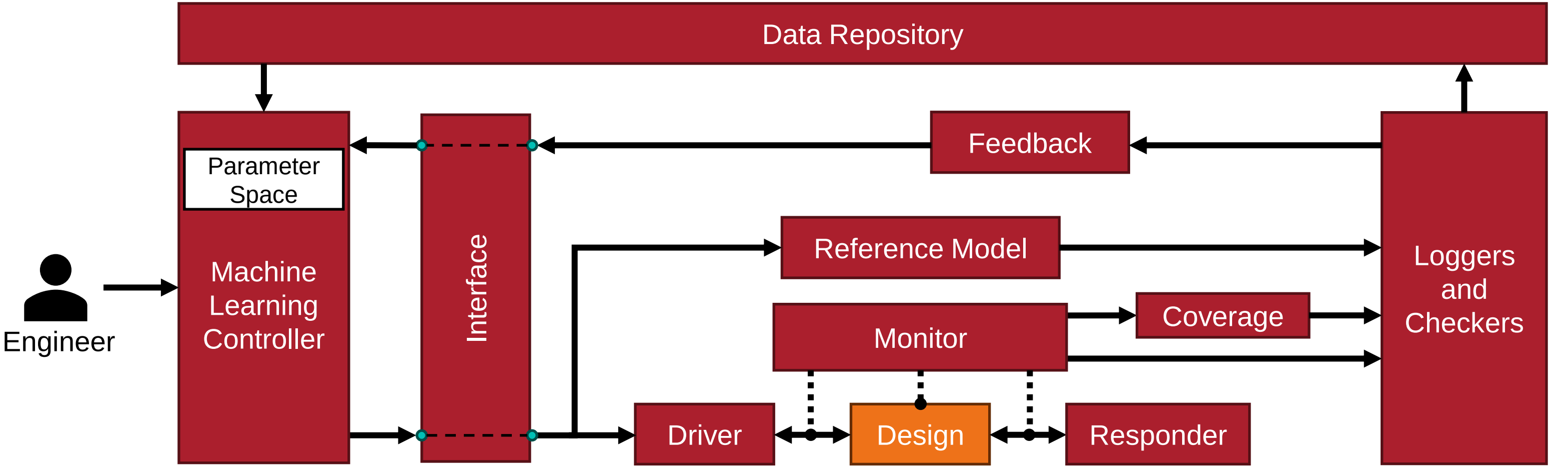
How ML modifies a typical testbench

- All research in this area aims to increase verification efficiency
- Engineers time, simulation time…



How ML modifies a typical testbench

- Human expert is now free to do other tasks?
- Research needed to make ML test-techniques easier to deploy and maintain

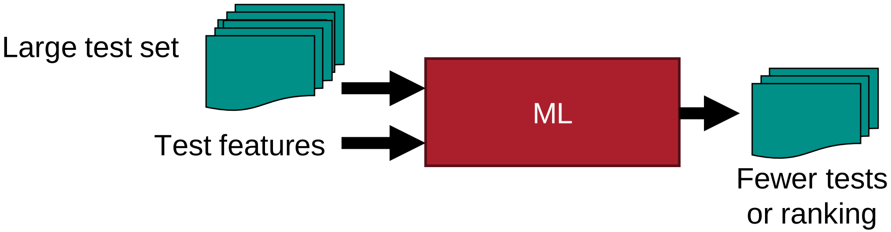


Classifying research by test controller

Test Generation: ML is trained to generate a test

Test Direction: the ML is trained to “direct” something else to generate the tests. Usually, by parameterising a constrained random test generator.

Test Selection: the ML chooses from pre-generated tests

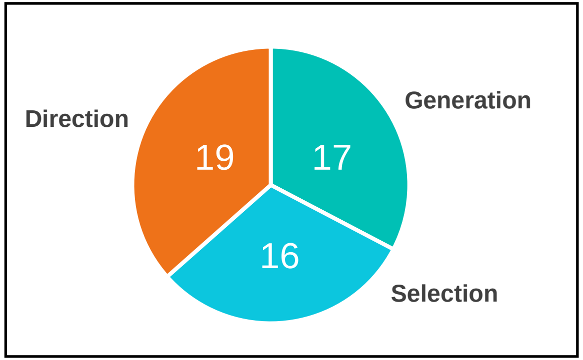


- **Set Optimisation** – choose a subset of tests, often offline
- **Test Filtering** – decide to simulate a test on a case-by-case basis, often online
- **Prioritisation** – decide order to run tests

Classifying research by test controller

- Compared to other domains, there is no "right" ML approach
- …then there's the choice of ML type (supervised, RL…) and algorithm (random forest, ANN, Q-learning….)

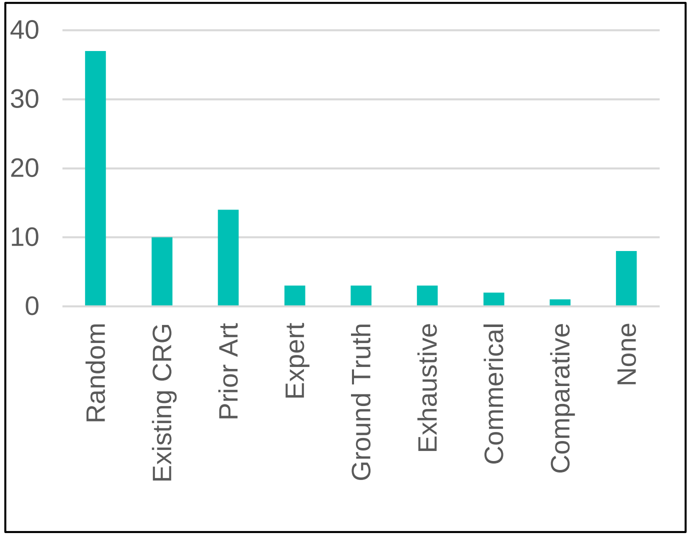
	Control over test content	Domain knowledge	Integration complexity
Generation	Direct	High	In the loop
Direction	In-direct	Low	In-direct
Selection	None	Low to None	None



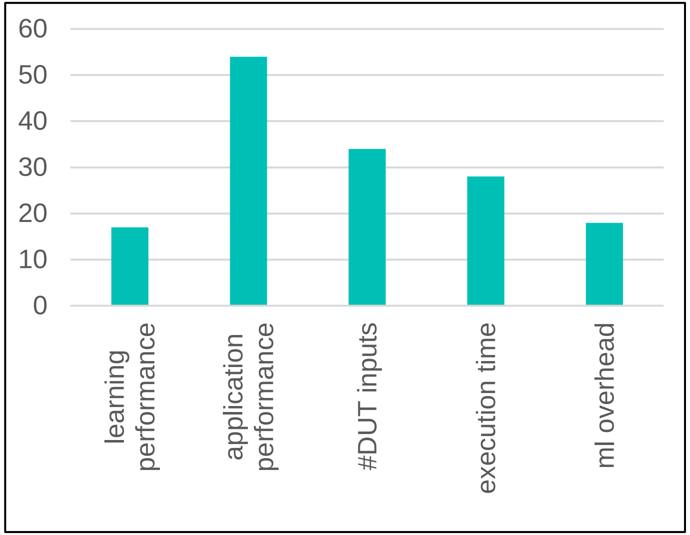
Number of papers by controller type

Evaluation baselines and metrics

Comparing techniques requires common baselines and metrics

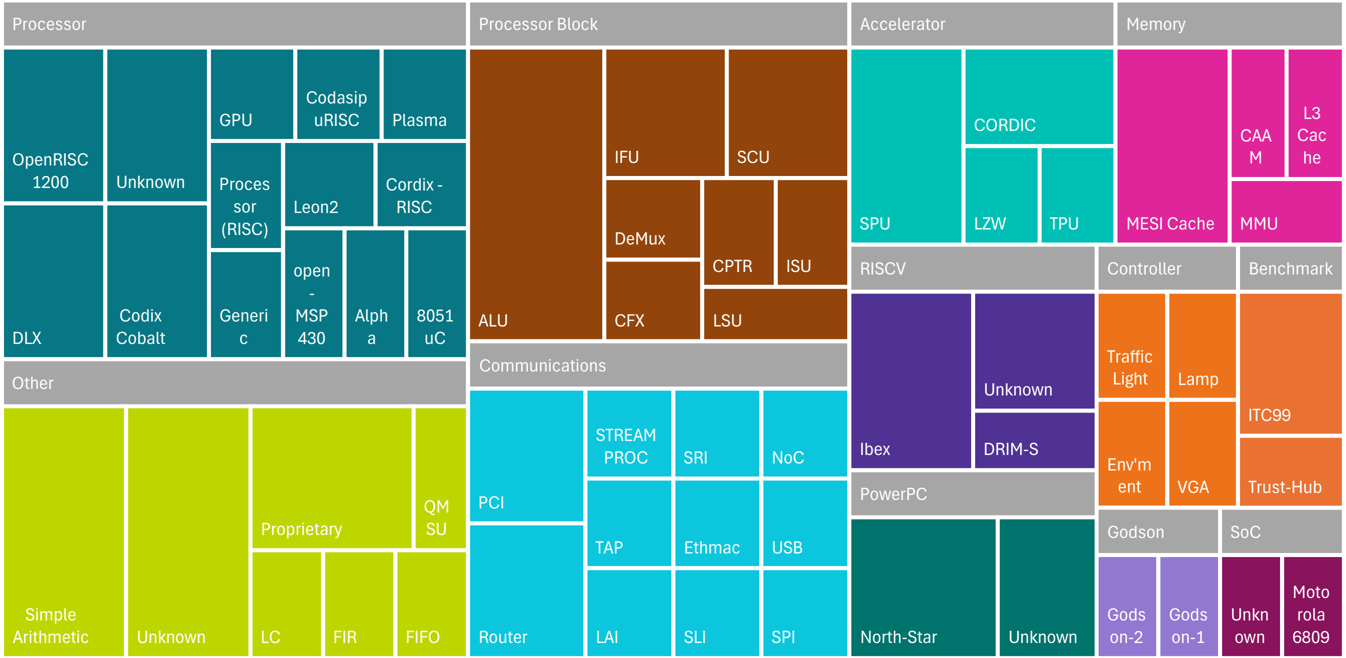


Baselines by appearances in research



Metrics by appearances in research

Evaluation designs



Designs used to evaluate techniques
(size of square indicates number seen in research)

- **Variety**: Shows ML test-techniques are widely applicable
- **Number**: reflects individual motivations and resources of research
- **Types**: Unknown, known function, open-source
- **Complexity**: varies

Lots of research but low adoption

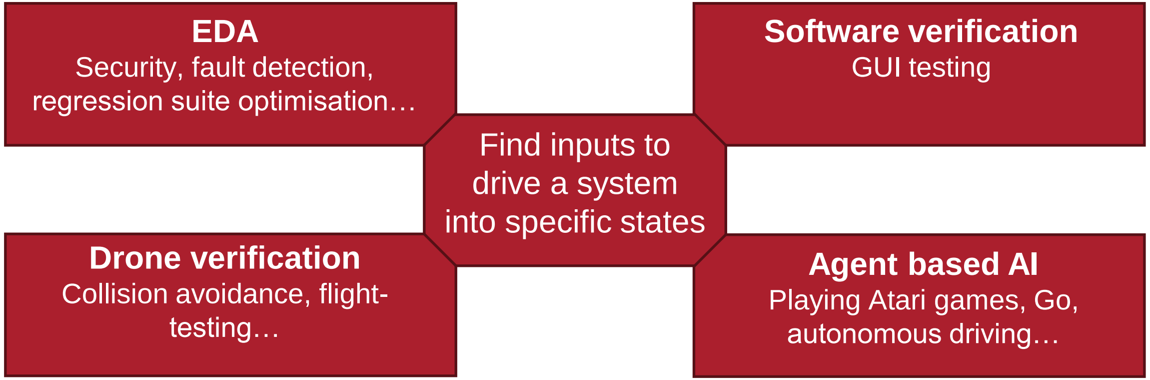
- **Reason one:** using ML to map coverage to test stimuli is difficult:
  - Lack of
    - positive training examples
    - distance metrics in coverage space and test space
    - enough positive training examples
  - Stimuli and micro-architecture behaviour at different abstraction levels
  - Design changes alter the ground truth relationships
- **Reason two:** too much choice has created a confusing landscape for practitioners -> classify research to aid adoption of ML techniques
- **Reason three**: high-level problem definition results in research effective at solving uniquely defined problems, but which is difficult to apply and complicates comparisons

Opportunities

- **Industry to set the agenda via problem and evaluation criteria**
  - Maturity and popularity of open designs (e.g., RISC-V)
  - Open datasets, pre-configured test benches and coverage models reduce barriers to entry
- **Researchers to provide better understanding to guide technique selection**
  - Classification is a first step
  - Quantification of resource-performance trade off needed

Opportunities

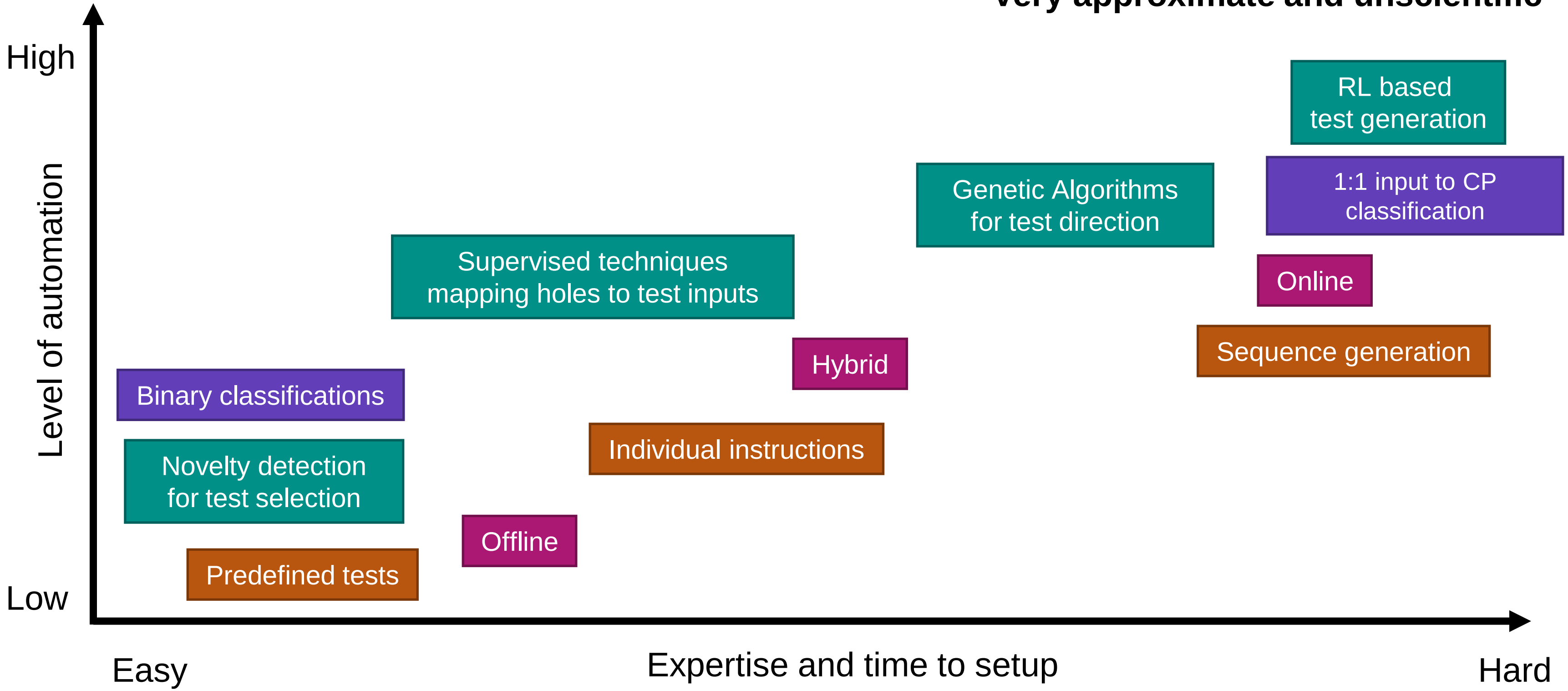
- **Look for synergies in the use of ML**
  - analogous findings for test-based **software verification**
  - similar applications in wider EDA and beyond



…and more!

What could be applied now vs future

Very approximate and unscientific



Summary

- 25 years of research material exists
- Classifications provide a means to navigate the options
- Industry can define the problem and evaluation criteria
- Research needed on which techniques to apply for a given design
- Look for synergies with the use of ML in wider EDA and beyond

Review of Machine Learning for Micro-Electronic Design Verification
https://arxiv.org/abs/2503.11687