Test and the Economics of Rare Defects

Module 8 : Test and Qualification | Lecture 1

Asif Khan
Associate Professor
School of Electrical and Computer Engineering
School of Materials Science and Engineering
Georgia Institute of Technology

Where this module sits in the course

Process	Structure	Yield	Test	Lifetime
deposition, etch, lithography	device geometry and materials	how many die are good	which die are good — today	how long they stay good — next time

The digital twin has covered everything up to yield. **This module adds the two steps that decide whether a product makes money: what the finished part measures, and how long it survives in the field.**

- **Lecture 1 —** test data, rare-defect screening, and the cost of getting the threshold wrong
- **Lecture 2 —** qualification, reliability physics, and extrapolating a ten-year claim from weeks of data

Two lectures, two different questions

1

Does this die work now?

Millions of measurements per wafer. Labels arrive within hours. The defective fraction is a few parts per million.

2

Will it still work in ten years?

A few dozen parts, stressed for a few weeks, with zero failures observed. The answer has to be extrapolated.

The two problems sit at opposite ends of the data spectrum.

	Lecture 1 — Test	Lecture 2 — Qualification
Sample size	10^8 – 10^{10} measurements per week	10 – 100 units per stress cell
Labels	Immediate and complete	Mostly censored, often zero events
Statistical regime	Extreme class imbalance	Extreme data scarcity
What ML supplies	Screening under imbalance	Priors, physics, and uncertainty

The organizing idea for today

Test is the largest labeled dataset in semiconductor manufacturing, and the class we care about is about one part in ten thousand.

Every method in this lecture exists because of that imbalance.

1

Rarity sets the method

Ordinary classifiers optimize the majority class. Here the majority class is uninteresting.

2

Rarity sets the metric

Accuracy and F1 are close to meaningless at 100 DPPM. Cost is the only honest scoreboard.

3

Rarity sets the payoff

Moving a screen by a fraction of a sigma changes revenue and warranty exposure by millions.

SEGMENT 1

Foundations: what test actually is

Enough vocabulary that the machine learning has something to refer to.

The test flow, and what data comes out of each stage

Wafer sort	Final test	Burn-in	System-level test
Probe each die on the wafer. Parametric and functional tests.	Test the packaged part over voltage and temperature.	Hold at elevated voltage and temperature for hours.	Run the part in a board that looks like the customer's system.
DATA EMITTED Per-die parameters, bin codes, X-Y location	**DATA EMITTED** Per-unit parameters, shmoo results	**DATA EMITTED** Pre- and post-stress parameter shifts	**DATA EMITTED** Pass/fail on real workloads

Cost per part rises sharply from left to right, and the ability to diagnose what went wrong falls. A defect caught at sort costs cents. The same defect found at system level test costs dollars. Found by the customer, it costs thousands.

Wafer sort: where most test data is born

- **Probe card.** A fixture with needles or spring pins that contacts the bond pads of one or more die at a time.
- **Touchdown.** One landing of the probe card. A modern card may test 64 or more die per touchdown.
- **Test program.** A sequence of applied stimuli and measurements, typically a few hundred to a few thousand per die.
- **Bin code.** The category the die falls into. Bin 1 is usually the best pass bin; other bins record the first test that failed.
- **X-Y coordinates.** Every record carries the die location on the wafer. Spatial structure is free information.

One 300 mm wafer

≈ 500–1000 die × ≈ 500–5000 measurements per die

≈ 10^6 numbers from a single wafer, and a fab runs tens of thousands of wafers per month.

≈10^6

measurements from one wafer

100 %

of die labeled pass or fail

After sort: packaged part, stress, and system context

1

Final test

The packaged part is measured again, across the voltage and temperature corners the product is specified for. Packaging itself introduces defects, so sort results do not carry over.

Adds: corner-by-corner parametric data

2

Burn-in

Parts are held at elevated voltage and temperature, often 125 °C for several hours, to force weak parts to fail before shipment.

Adds: parameter shift before vs. after stress

3

System-level test

The part runs realistic workloads on a system board. It catches failures that only appear under real traffic, real power delivery, and real thermal conditions.

Adds: pass/fail under conditions no test program models

Parametric test: the vector a machine learning method sees

- **Parametric tests** measure a continuous physical quantity: a current, a voltage, a resistance, a delay, a frequency.
- **E-test, or in-line parametric test,** measures dedicated structures in the scribe lines between die — transistor threshold voltage, contact resistance, line capacitance.
- **Each die therefore has a vector** of a few hundred to a few thousand real numbers, plus its coordinates and its lot and wafer identity.

This vector is the input to almost every method in the next segment.

Nothing about the methods is specific to semiconductors. What is specific is the structure: strong correlation between parameters, drift between lots, and spatial correlation across a wafer.

ONE DIE RECORD

Lot / wafer / die X, Y	**K2431 / 07 / (14, 22)**
Idd_standby	**4.82 µA**
Vth_nmos	**0.312 V**
Ring oscillator freq.	**2.914 GHz**
Contact resistance	**18.4 Ω**
Leakage @ 1.1 V, 85 °C	**1.07 mA**
…	~1200 more tests
Bin code	**1 (pass)**

How the data is stored: the STDF record format

- **Standard Test Data Format** is the file format testers write. It has been the industry standard since the late 1980s.
- **It is a binary stream of typed records,** not a table. Reading it means walking the stream and reconstructing the hierarchy.
- **Practical consequence:** the first real task in any test-analytics project is parsing and flattening STDF into a dataframe, and it is not trivial.

Open-source parsers exist in Python and Rust. Budget real time for this step anyway: files are large, vendors extend the format, and per-site and per-touchdown metadata is easy to lose.

Record	What it holds
MIR / MRR	Master information: lot, product, program, start and end time
WIR / WRR	Wafer information: wafer ID, number of parts tested and passed
PIR / PRR	Part information: one pair per die, with X-Y location and final bin
PTR	Parametric test result: test number, measured value, limits, pass flag
FTR	Functional test result: pass or fail for a pattern-based test
TSR	Test synopsis: per-test summary statistics for the whole lot

Two ways to test a chip

Functional test

Apply realistic inputs and check that the outputs are right. Directly answers the question the customer cares about.

- Requires knowing what correct behaviour looks like
- Coverage of internal nodes is hard to quantify
- Test time grows quickly with design size
- Still essential at system-level test

Structural test

Ignore what the chip is for. Treat it as a netlist of gates and check that each gate and wire behaves.

- Coverage is measurable against a defect model
- Patterns are generated automatically
- Test time is far shorter for the same confidence
- Dominates modern digital test

Structural test is what makes the data in this lecture exist. Automatic pattern generation, measurable coverage, and machine-readable failure signatures all come from treating the chip as a netlist.

Scan and automatic test pattern generation

- **Scan design.** Every flip-flop in the chip gets a second input. In test mode all the flip-flops connect into long shift registers called scan chains.
- **What that buys.** Any internal state can be shifted in, one clock cycle of real logic applied, and the resulting state shifted back out. Deep internal nodes become directly controllable and observable.
- **Fault models.** A defect is abstracted as, for example, a node stuck at logic 1, a node stuck at logic 0, a transition that is too slow, or a bridge between two nodes.
- **ATPG.** Automatic test pattern generation software solves, for each modeled fault, for an input pattern that makes that fault visible at an output. It then compacts the patterns so that one pattern covers many faults.
- **Coverage.** The fraction of modeled faults that at least one pattern detects. Production programs target well above 99 percent for stuck-at faults.

WHY THIS MATTERS HERE

Scan makes the chip observable, and observability is what turns manufacturing into a data problem.

When a part fails a scan pattern, the tester records exactly which scan cells captured the wrong value. That failure signature is rich enough to reason backwards to a physical location on the die.

We come back to this in the third segment, where volume scan diagnosis turns those signatures into a ranked list of what is going wrong in the fab.

Memory self-test and on-die monitors

Memory built-in self-test (MBIST)

Memory arrays are too dense and too regular for external patterns to be efficient. A small hardware engine is built onto the die that writes and reads algorithmic patterns through the array by itself.

- Runs at full speed, with no tester bandwidth needed
- Reports failing address and failing bit
- Drives repair: spare rows and columns are swapped in by blowing fuses
- **Data value:** a bit-level failure map of every array on the die

On-die monitors and IJTAG

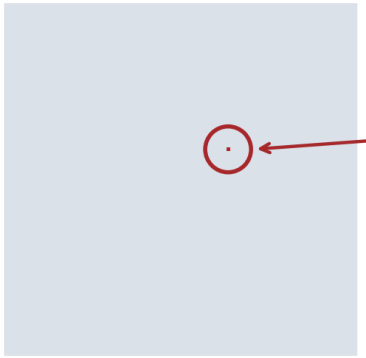
Small sensors are placed around the die: ring oscillators for speed, thermal diodes, voltage droop detectors, aging monitors. IJTAG (IEEE 1687) is the standard network for reaching them.

- Measures local conditions rather than chip-average conditions
- Readable on the tester and also in the field
- **Same instruments, two uses:** screening today, aging telemetry for years afterwards
- This is the hardware that makes Lecture 2's lifecycle story possible

Both of these put measurement hardware on the die itself: the chip increasingly reports on its own condition, on the tester and afterwards.

The framing that governs everything else

Enormous label volume, and almost no positive labels.



10,000 parts shown here.
100 DPPM means 1 in 10,000.
That one part is the whole learning problem.

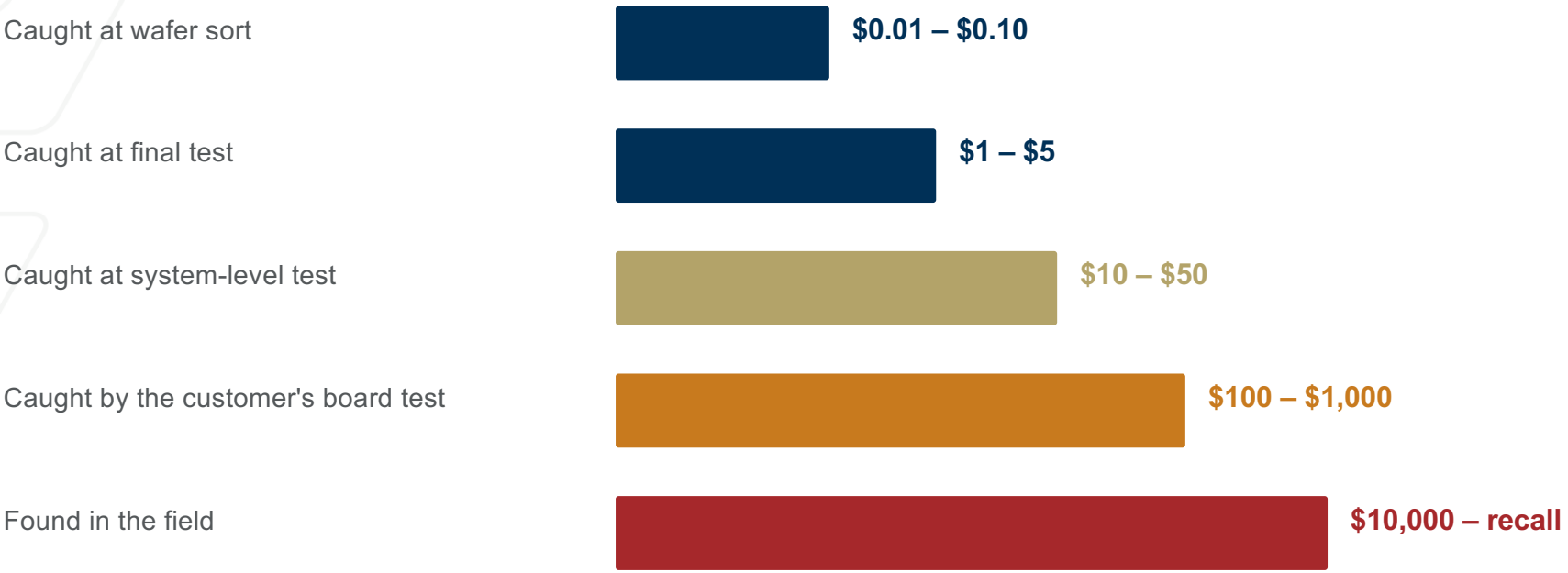
DPPM

Defective parts per million.

The number of parts out of every million shipped that turn out to be bad. It is the currency of quality in this industry.

Market	Typical DPPM target
Consumer electronics	500 – 1000
Data center and enterprise	50 – 200
Automotive (AEC-Q100)	< 10, with zero-defect as the stated goal
Aerospace and medical	Effectively zero, verified part by part

Why a single escaped part is expensive



Each stage downstream multiplies the cost of the same defect by roughly ten. This is the reason a screen that throws away good die can still be profitable.

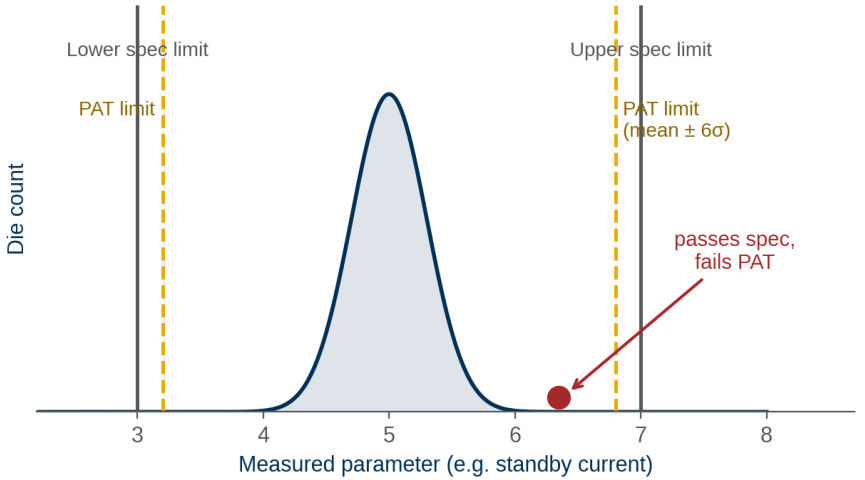
SEGMENT 2

Outlier screening

The intellectual core of the lecture. Everything here is a response to rarity.

The problem: a die can pass every test and still be bad

- **Spec limits are set by the product requirement,** not by what the process normally produces. They are usually wide.
- **A latent defect often shifts a parameter slightly** without pushing it outside the spec. The part passes.
- **That part then fails early in the field,** because the same defect that shifted the parameter also degrades under stress.



The key move

Stop asking whether a die is inside the spec. Ask whether it looks like its peers.

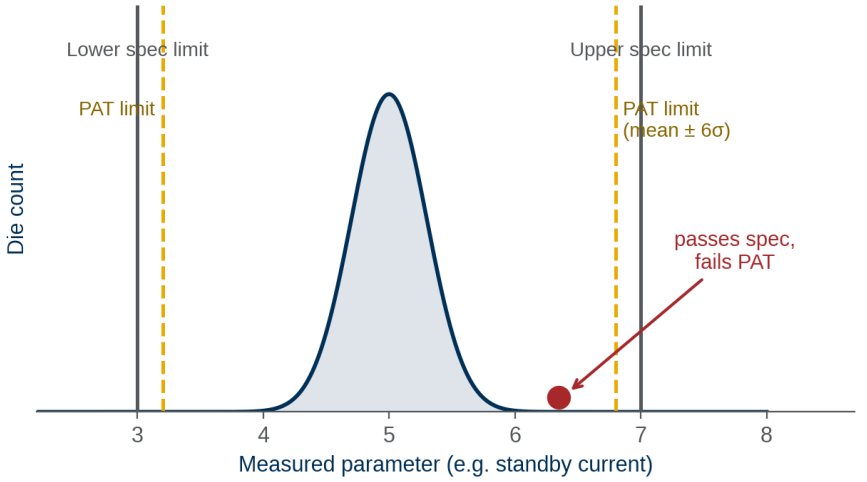
CLASSICAL PROGRESSION, STEP 1

Part average testing

- **The rule.** Compute the mean and standard deviation of a parameter across a population of die. Fail any die outside mean ± 6σ, even if it passed the spec limit.
- **Where it came from.** The automotive supply chain. AEC and the Automotive Industry Action Group published it as a required practice in the late 1990s.
- **Why 6σ.** It is a convention, chosen so that the screen removes a small fraction of a well-behaved population. It is not derived from cost.

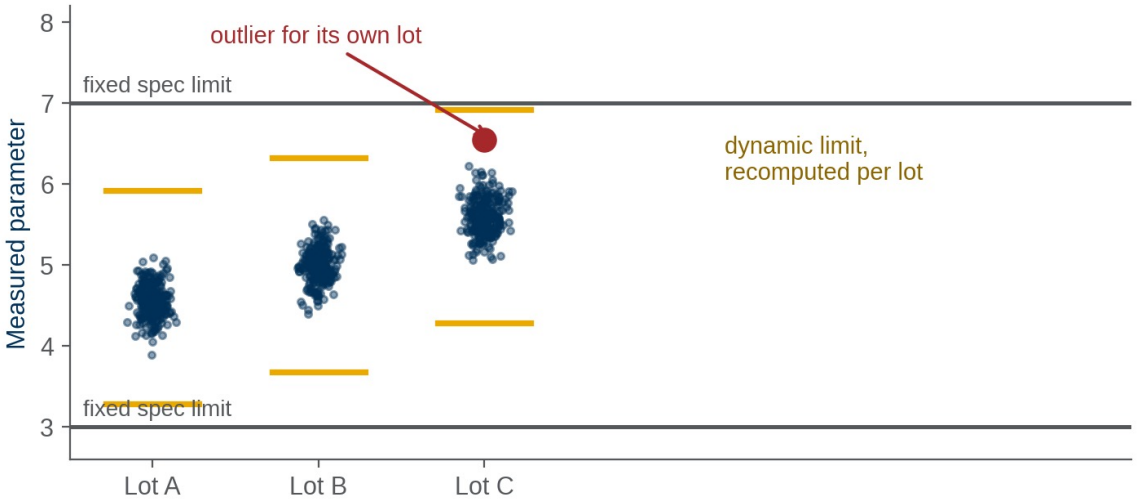
Limitations that drive the next three slides

Mean and standard deviation are themselves distorted by the outliers you are hunting. And a fixed limit computed once does not follow real process drift.



CLASSICAL PROGRESSION, STEP 2

Dynamic part average testing



- **The change.** Recompute the limits for each wafer or each lot, from that wafer's own population.
- **Why it works.** Process conditions drift. A value that is ordinary in one lot can be extreme in another.

WHAT THE PICTURE SHOWS

Three lots of the same product. The population centre moves from lot to lot because the process moved.

The grey lines are the fixed spec limits. They never move, and they never fire.

The gold lines are recomputed for each lot. The red die is ordinary against the fixed limit and clearly abnormal against its own lot.

Choosing the population is now a modelling decision: per wafer, per lot, per test site, or per equipment chamber.

CLASSICAL PROGRESSION, STEP 3

Robust statistics: z-PAT

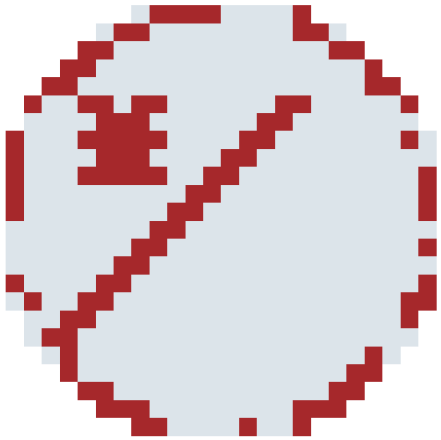
- **The problem.** The sample mean and sample standard deviation are pulled toward the outliers. A few extreme die inflate σ, the limits widen, and the screen loses sensitivity. The outliers hide themselves.
- **The fix.** Use estimators that ignore the tails. Replace the mean with the median. Replace the standard deviation with a robust scale estimate.

Estimator	Definition	Breakdown point
Mean, standard deviation	Sum-based	0 % — one bad point moves it
Median	50th percentile	50 %
IQR / 1.35	Interquartile range, scaled to σ	25 %
Median absolute deviation × 1.4826	median($\lvert x_i - \text{median}(x)\rvert$), scaled to σ	50 %

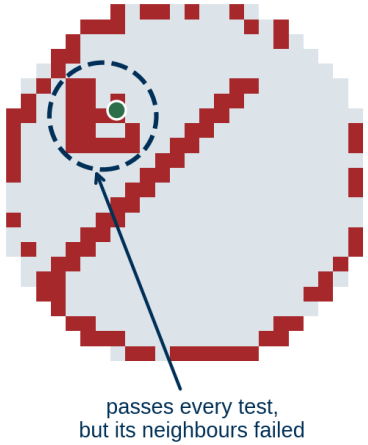
Breakdown point: the fraction of the sample that can be arbitrarily corrupted before the estimator becomes arbitrarily wrong.

Spatial methods: the wafer map carries information

Pass / fail wafer map



Good die in a bad neighbourhood



- **Defects are not spatially random.** Scratches, edge effects, particles and chamber signatures all produce structure.

NEAREST-NEIGHBOUR RESIDUAL

For each die, predict its parameter value from its neighbours. Screen on the residual rather than on the raw value.

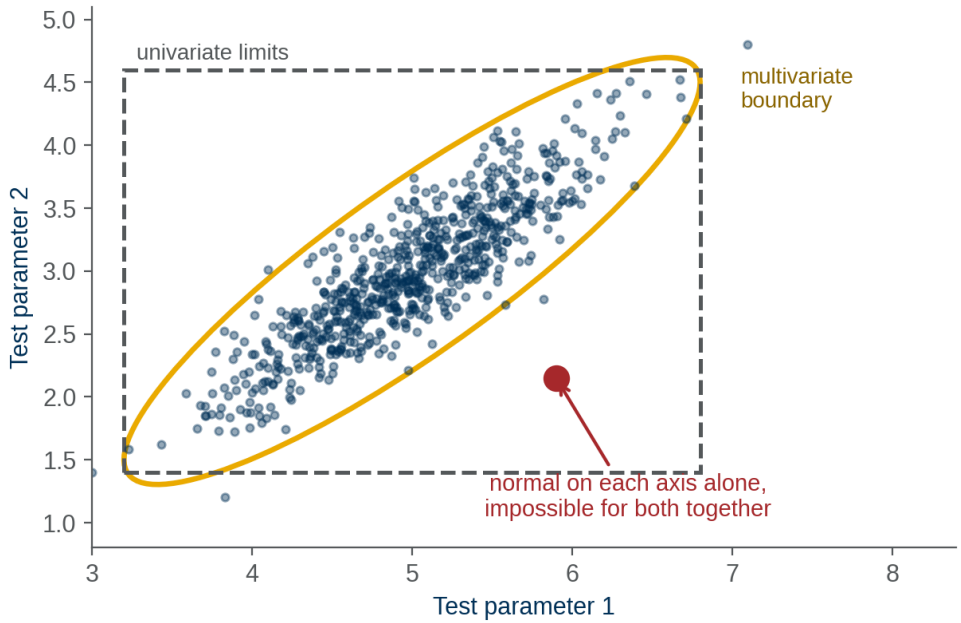
$r_i = x_i -$ median(x over the neighbours of i)

This removes smooth wafer-scale variation, which is normal, and leaves the local anomalies, which are suspicious.

Good die in a bad neighbourhood

A die that passes every test while its immediate neighbours failed is scrapped on location alone. The reasoning is that whatever damaged the neighbours probably touched it too.

Multivariate outlier detection



- **Parameters are correlated.** They are driven by the same underlying process variation, so they move together.
- **Testing one axis at a time discards that.** A die can be unremarkable on every single parameter and still occupy a combination the process never produces.
- **The generalization is direct.** Score each die by how unlikely its whole vector is under the population, and threshold the score.

Mahalanobis distance — the simplest version

$$d^2(x) = (x - \mu)^T \Sigma^{-1} (x - \mu)$$

Euclidean distance after whitening by the covariance. Robust versions estimate μ and Σ with the minimum covariance determinant method.

The method menu, and what each one assumes

Method	Assumption about normal data	Fits this problem when
Robust z-score per test	Each parameter is unimodal	Interpretability and a per-test audit trail are required
Mahalanobis / robust MCD	One elliptical cluster	Parameters are correlated and roughly Gaussian
PCA reconstruction error	Data lies near a low-dimensional linear subspace	Hundreds of highly redundant parameters
Isolation forest	Anomalies are easy to separate by random splits	No distributional assumption is safe
One-class SVM / SVDD	A compact region contains the normal class	The boundary is nonlinear but the sample is moderate
Autoencoder reconstruction error	A nonlinear manifold captures normal data	Very large samples and nonlinear structure
Gaussian mixture / clustering	Several distinct normal modes exist	The process genuinely runs in multiple states

All of these are unsupervised. They are trained on the passing population and score how unusual a die is. None of them are trained on labelled escapes, because escapes are too rare to train on.

Escape and overkill

Escape

A defective part that passed the screen and shipped. Measured in DPPM. Discovered weeks or months later, by the customer.

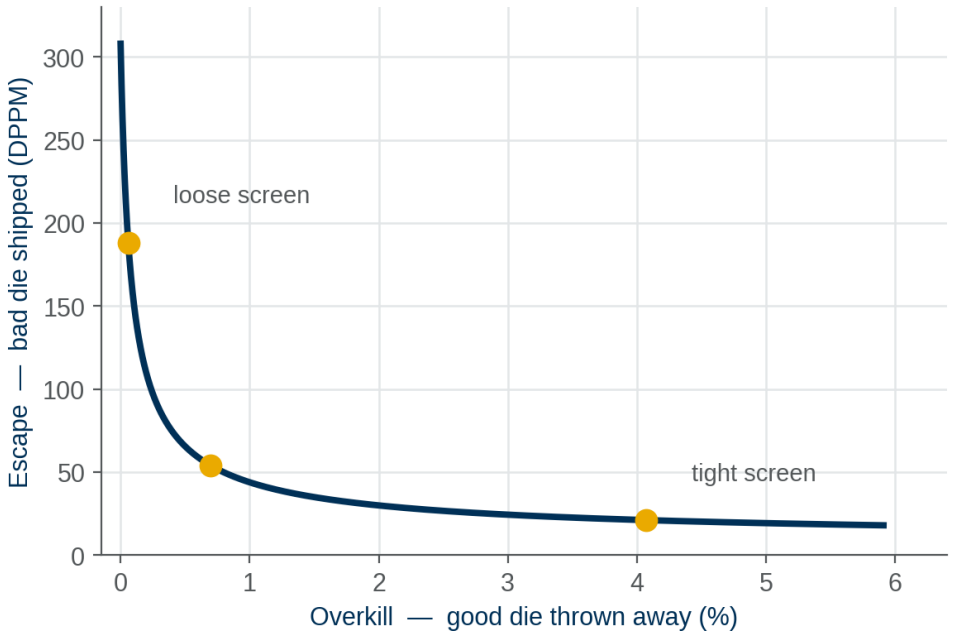
Overkill

A good part that the screen threw away. Measured in percent of yield. Discovered immediately, on your own yield report.

Tightening the screen always trades one for the other. There is no setting that reduces both.

	Escapes	Overkill
Who pays	The customer, then you	You, immediately
When it is visible	Weeks to months later	The same shift
How it is measured	Returns, field failures, DPPM	Yield loss, percent
Typical unit cost	$100 – $10,000+	$1 – $50
Political pressure inside a company	Diffuse and delayed	Immediate and loud

The escape–overkill trade-off curve



- **Sweep the threshold,** and plot the two error rates against each other. Every point on the curve is a screen you could actually deploy.
- **The shape is the whole story.** Near the loose end, a small amount of overkill buys a large reduction in escapes. Near the tight end, large amounts of overkill buy almost nothing.
- **This is a receiver operating characteristic** in the units the business uses. The axes are DPPM and yield percent, not true and false positive rate.
- **A better model moves the whole curve down and left.** Comparing two methods means comparing curves, not

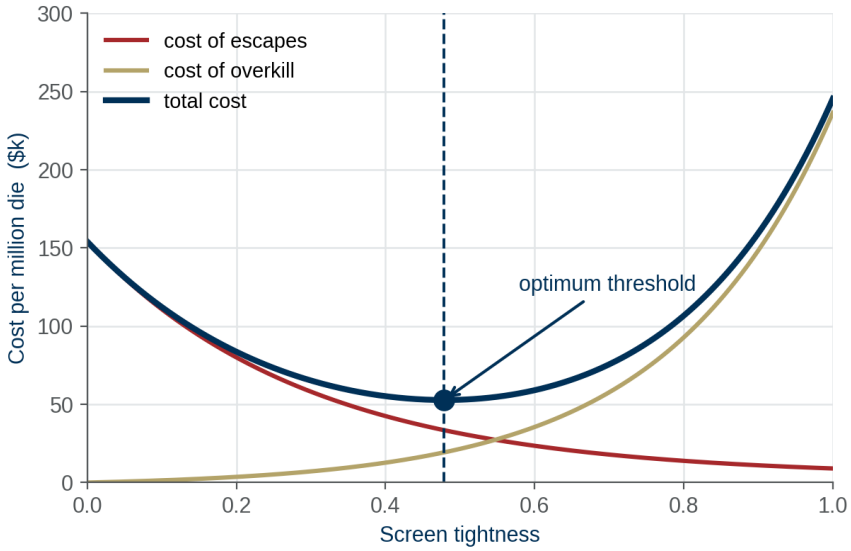
The curve says what is possible. It does not say where to sit. That takes a cost function.

Choosing the operating point: write the cost function down

Total cost per million die, as a function of the threshold τ

$$C(\tau) = N \cdot E(\tau) \cdot c_{escape} + N \cdot O(\tau) \cdot c_{overkill}$$

- **E(τ)** — escape rate at threshold τ, from the curve on the previous slide
- **O(τ)** — overkill rate at threshold τ, from the same curve
- **c_escape** — fully loaded cost of one escaped part: return, analysis, replacement, customer credit, and reputational exposure
- **c_overkill** — cost of scrapping one good die: accumulated wafer cost divided by die per wafer, plus lost margin
- **Then minimise.** τ* = argmin C(τ). One line of code once the curve exists.



Why accuracy and F1 are the wrong metrics here

One million parts. 100 are defective. Compare two screens.

	Screen A — passes everything	Screen B — a real screen
Defects caught	0 of 100	85 of 100
Good die scrapped	0	4,000
Accuracy	99.99000 %	99.59915 %
Precision	undefined	2.1 %
Recall	0 %	85 %
F1 score	0	0.041
Cost at $500 / escape, $4 / good die	**$50,000**	**$23,500**

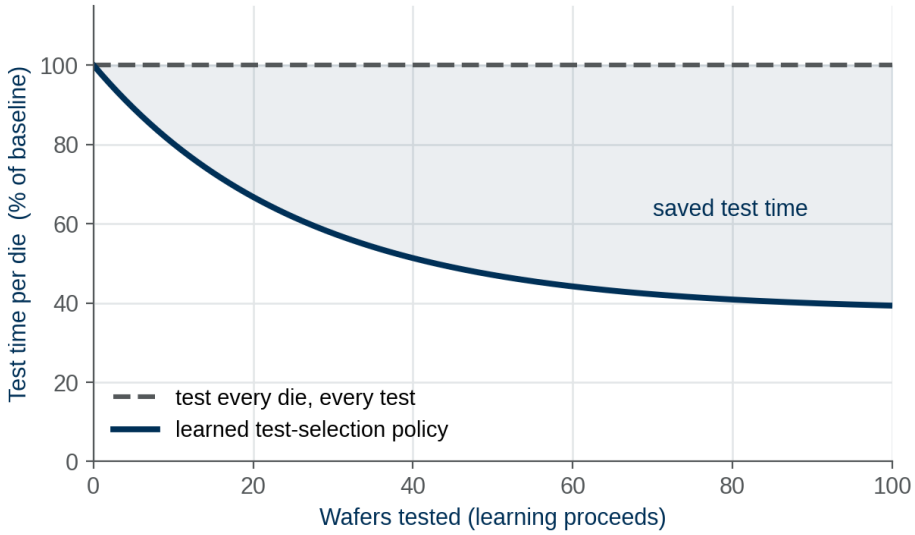
Screen A wins on accuracy. Screen B wins on the only number that is real. Report the trade-off curve and the cost, and treat every classification metric as a diagnostic.

SEGMENT 3

Adaptive test and machine learning inside DFT

Using the same data to decide what to test, and to improve the test itself.

Adaptive test: choose what to test, per part

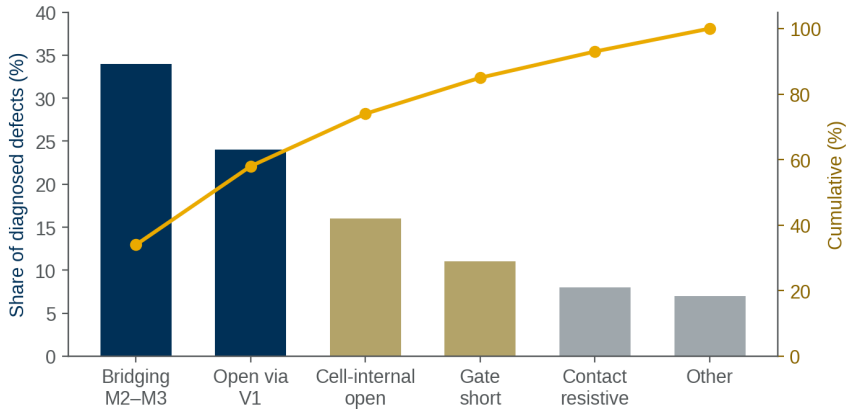


- **Test selection.** Drop tests that have never failed on this product, or whose outcome is predictable from tests already run.
- **Test ordering.** Run the tests most likely to fail first, so failing parts stop early and release the tester.
- **Early termination.** Stop testing a part as soon as its outcome is decided with sufficient confidence.
- **Adaptive routing.** Send only the parts a model flags as risky to burn-in or to system-level test.
- **Feedback across stages.** Use wafer sort results to set the final-test program for that same die.

Test cost is a large share of the cost of a finished part. Cutting test time is one of the few levers that shows up directly in gross margin.

Volume scan diagnosis and yield learning

- **Start from the failure signature.** When a part fails a scan pattern, the tester records which scan cells captured the wrong value.
- **Diagnosis inverts the netlist.** Software searches for the physical net or cell whose failure would explain the observed signature.
- **Cell-aware diagnosis goes inside the cell.** It uses transistor-level models of each standard cell, so it can name a defect inside a gate rather than only on a wire.
- **Volume diagnosis aggregates.** Run this over thousands of failing parts and the result is a statistical picture of what the fab is doing wrong.



The output is a defect Pareto: a ranked list of defect mechanisms with an estimated share of total yield loss. It tells process engineers where to spend their next month.

- **Where learning enters:** resolving ambiguous diagnoses, clustering signatures into mechanisms, and correlating the resulting Pareto against equipment and process logs.

Machine learning applied to the test hardware itself

1

Testability prediction with graph neural networks

A netlist is a graph. A GNN trained on netlists predicts which nodes will be hard to control or observe, early enough in design to fix cheaply.

Research, moving to industry

2

Test-point insertion with reinforcement learning

Extra observation and control points raise coverage and cost area. Placing them well is a sequential decision problem, and RL agents now beat classical heuristics on benchmarks.

Active research

3

Pattern set compaction

The generated pattern set is redundant. Learned ordering and selection shrink it at equal coverage, which cuts test time directly.

In production use

4

Alternate test for analog and RF

Specification tests for RF parts need expensive instruments. Measure cheap signatures instead and regress them onto the expensive specifications. Founded at Georgia Tech by Abhijit Chatterjee.

Established, still evolving

You have already been doing this

The WM-811K wafer map lab was a test analytics problem the whole time.

- **A wafer map is test data.** Each pixel is a die, and its colour is the bin code that came out of wafer sort.
- **Pattern classification is spatial screening.** Naming a map as a scratch, a ring, or a centre pattern is exactly the spatial reasoning in the good-die-in-a-bad-neighbourhood rule.
- **The imbalance was there too.** Most maps are unpatterned, and the interesting classes are rare.

WHAT CHANGES NOW

You were classifying maps. From today you are also deciding what to do about them, and that decision has a price attached.

The wafer map told you a pattern exists. The screening question is which individual die to scrap because of it, and the answer depends on what an escape costs you.

Summary

1. **Test is a very large labeled dataset with a very rare positive class.**
   That single fact determines the methods, the metrics, and the payoff.

2. **Screening asks whether a die resembles its peers, not whether it meets spec.**
   PAT, dynamic PAT, robust statistics, spatial residuals, and multivariate scoring are one idea at increasing generality.

3. **Every screen is a point on an escape–overkill curve.**
   A better model moves the curve. Choosing where to sit on it needs a cost function.

4. **Classification metrics rank screens incorrectly at these rates.**
   Write the costs down and minimise total cost. That is the honest objective.

5. **The same data supports deciding what to test, and diagnosing the fab.**
   Adaptive test cuts cost; volume diagnosis returns information to the process.

The exercise

Build the escape–overkill curve on a wafer-sort parametric dataset, then optimise the threshold against a cost function you write down.

1

Load and explore

Parametric vectors with die coordinates. Look at the correlation structure and the lot-to-lot drift before modelling anything.

2

Build two screens

A robust per-test z-score screen, and one multivariate screen of your choice from the menu.

3

Sweep and plot

Produce the escape–overkill curve for each screen. Compare the curves, not single points.

4

Write the cost function

State c_escape and c_overkill explicitly and justify them. Find the optimal threshold for each screen.

5

Report

Which screen wins, at what cost, and how the answer changes if the cost ratio changes by ten times in each direction.

The deliverable is the cost analysis, not the classifier. A simple screen with an honest cost argument scores higher than a complicated screen without one.