ITC 2021
Silicon Lifecycle Management Workshop

Training in Turmoil:
Silent Data Corruption in Systems at Scale

Rich Bonderson
October 14, 2021

“bugs-from-hell”

Application Users and Software Developers are CONFUSED

They are dealing with mysterious, difficult to identify problems.

Their hardware is abstracted away from them, in very real physical ways,
which is especially true of newer cloud-centric system architectures.

They are generally used to the assumption that hardware is good.

But more often now, especially at larger scales,
they’re encountering a hidden scourge – **Silent Data Corruption**.

A shift in mindset and a need to react and prevent is here.

How much effort will be needed to mitigate and make these changes?

Expanding the Software Story

Encountering non-deterministically wrong answers.

Problems can appear in a different iteration of a program,
acting on different data, or in a different system scenario.
Problems reproduce at varying rates.

With notions that there is an underlying issue, SW can detect mismatches,
but this requires running multiple times, debug, and manual interventions.

Additionally, in some cases of ML models,
incorrect data will not cause failures.

This may sometimes be fine,
but must be minimized and carefully managed.
We need to stay ahead of this problem!!

TPU Hardware Systems

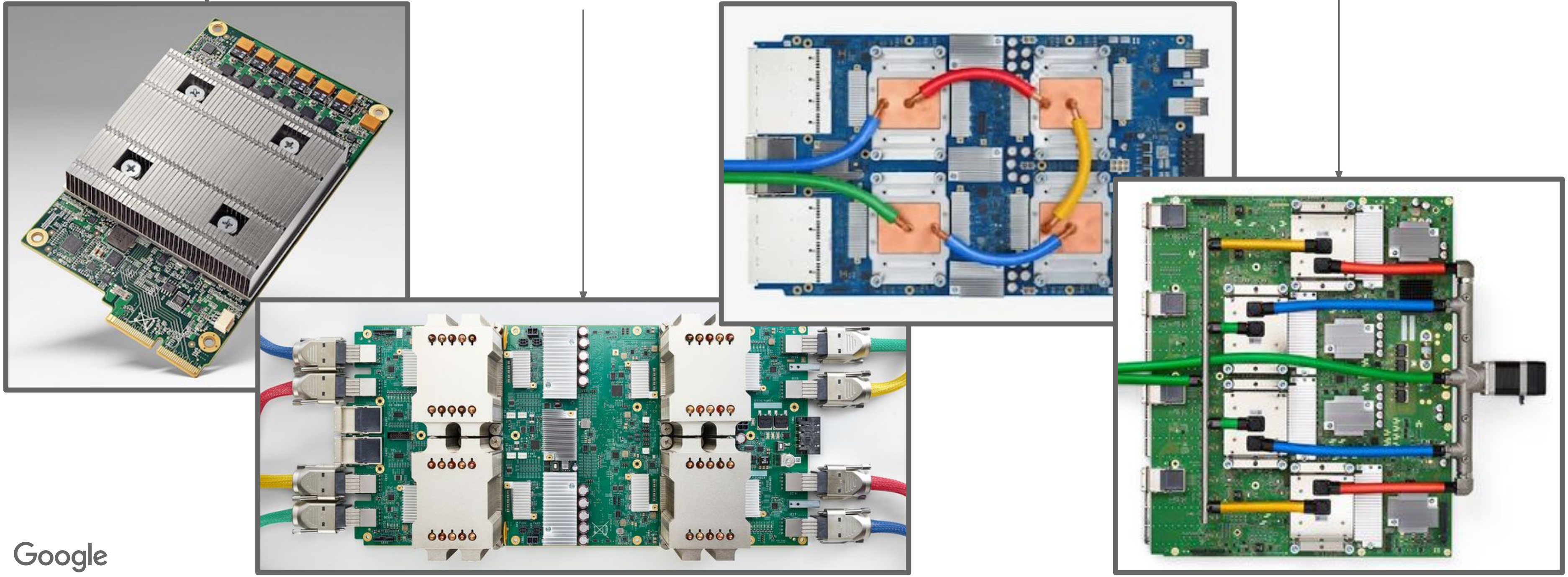
TPU Scaling

TPUv1, 2016

TPUv2, 2017
180 teraflops
64GB HBM

TPUv3, 2018
420 teraflops
128GB HBM

TPUv4, 2021
>840 teraflops



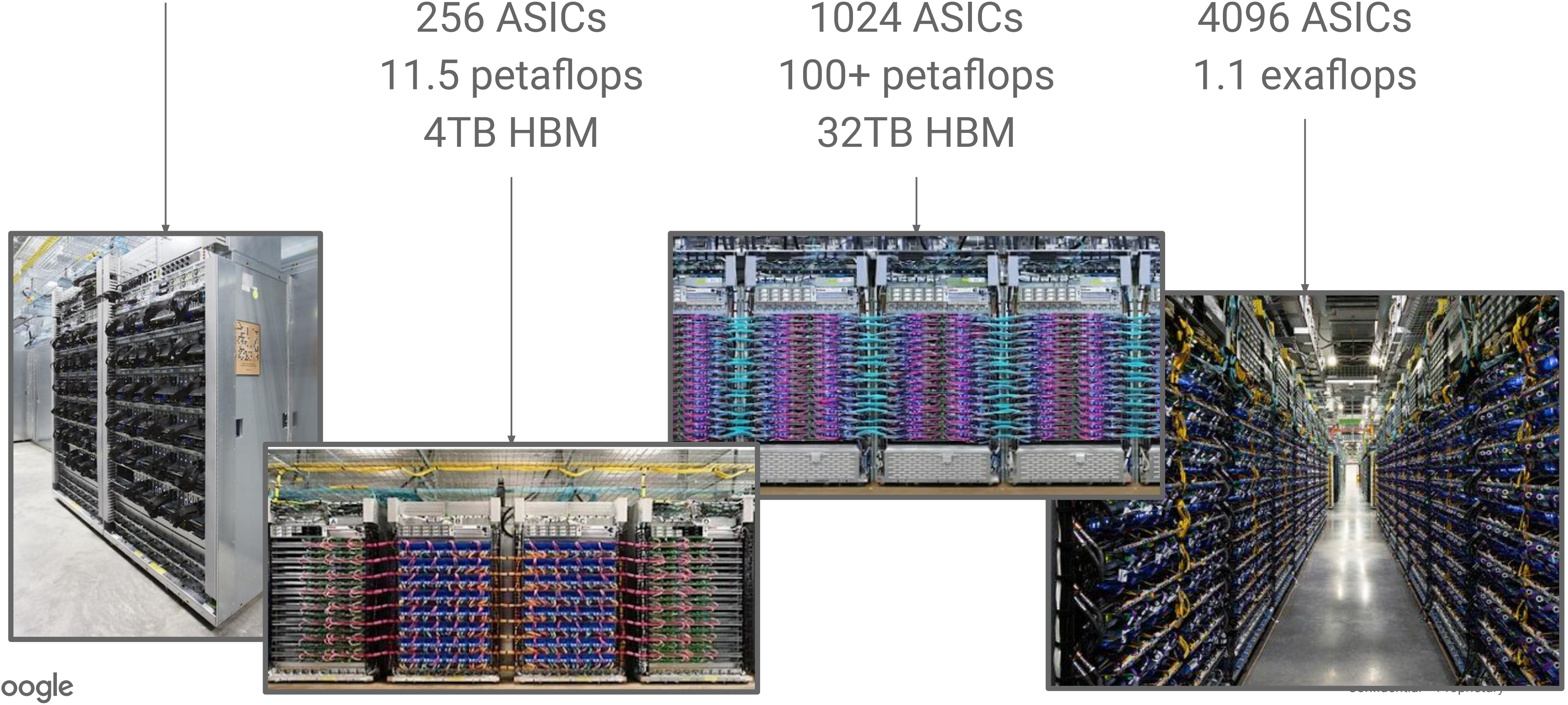
TPU Pod Scaling

TPU, 2016

TPUv2, 2017
256 ASICs
11.5 petaflops
4TB HBM

TPUv3, 2018
1024 ASICs
100+ petaflops
32TB HBM

TPUv4, 2021
4096 ASICs
1.1 exaflops



Research with TPUv4 Scale

*“Though the margin of difference in topline MLPerf benchmarks can be measured in mere seconds, this can translate to **many days worth of training time on the state-of-the-art models that comprise billions or trillions of parameters**.*

*To give an example, today we can train a **4 trillion parameter dense Transformer with GSPMD on 2048 TPU cores**. For context, this is over 20 times larger than the GPT-3 model published by OpenAI last year. **We are already using TPU v4 Pods extensively within Google to develop research breakthroughs such as MUM and LaMDA**, and improve our core products such as Search, Assistant and Translate.”*

1.1 exaflops * 10 days = ~1e24 flops.

That, folks, is a ***YOTTA*** flops!

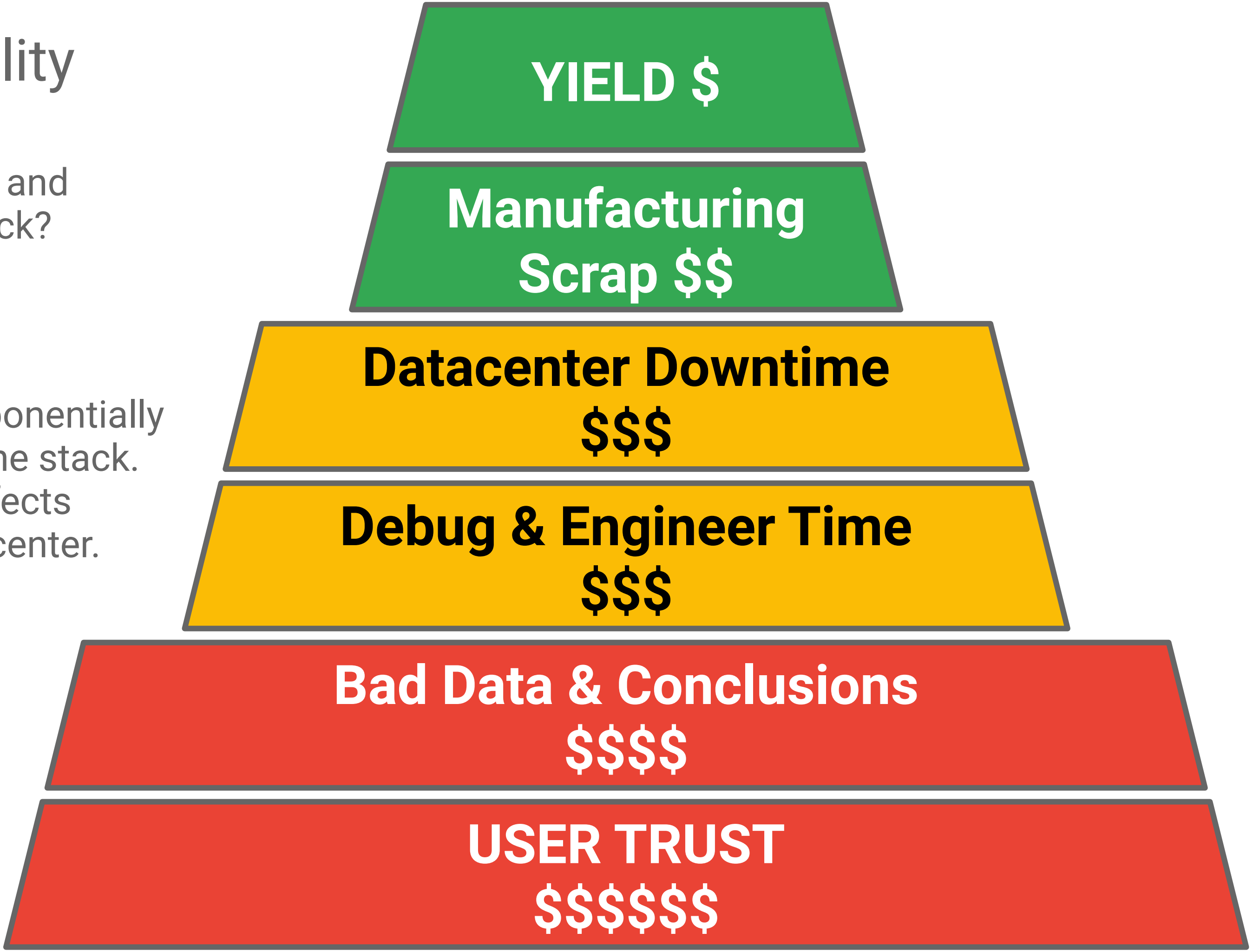
What happens when one goes bad?
What if you don’t know if it happened?

The Cost of Quality

How do we keep defects and SDC to the top of the stack?

Activity ANYWHERE has cost implications!

However, cost grows exponentially with time deeper down the stack. We need to eliminate defects before entering the datacenter.



YIELD $

Manufacturing Scrap $$

Datacenter Downtime $$$

Debug & Engineer Time $$$

Bad Data & Conclusions $$$$

USER TRUST $$$$$$

How’d we get here?

Initial Detection Mechanisms - NaN

In some of our earliest modern experience with this, “Silent” Data Corruption became not-so-silent with unexplained NaN faults - Not a Number.

What is NaN?

IEEE has a few special encodings

exp = 0xff, man = 0x0 → ±∞

exp = 0xff, man != 0x0 → NaN

0x77c00000 ~ 7.7884452878e+33, but 0x7fc00000 is NaN

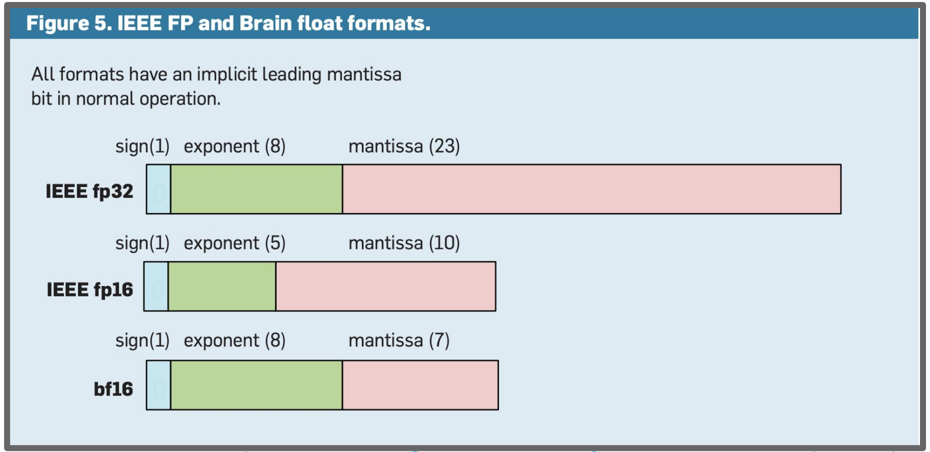


Figure 5. IEEE FP and Brain float formats.

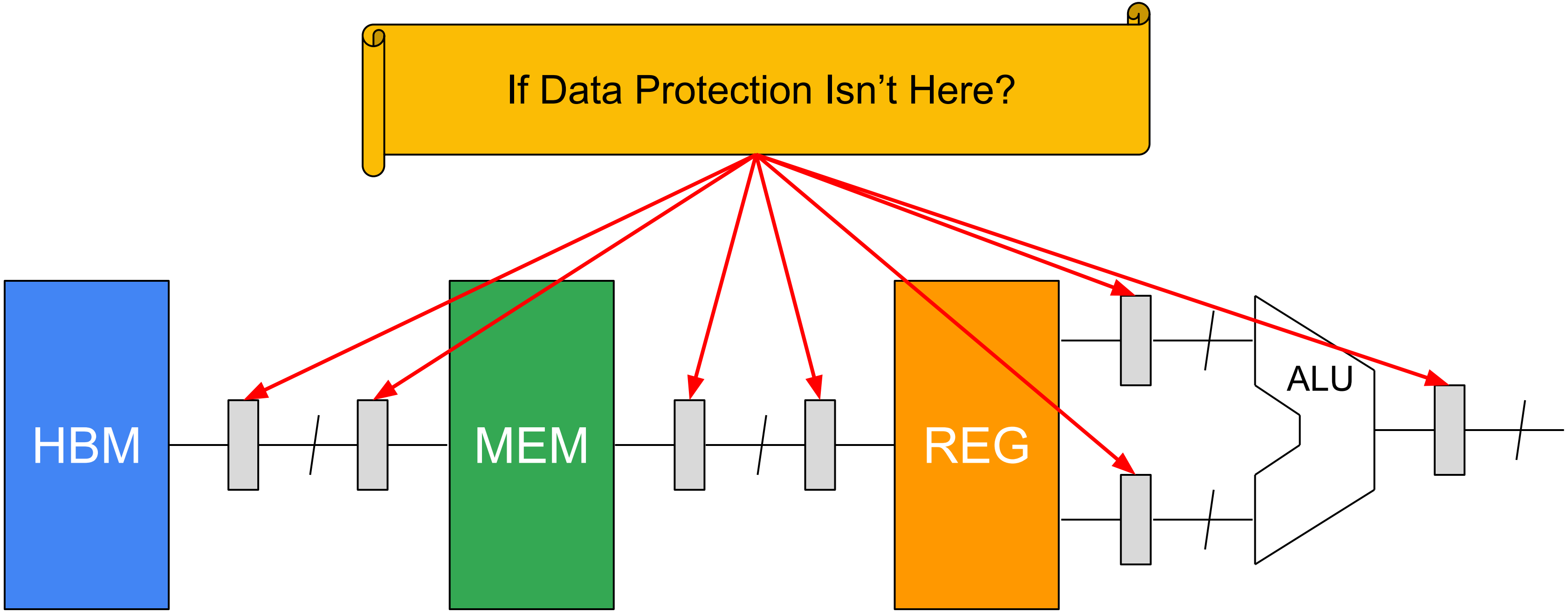
All formats have an implicit leading mantissa bit in normal operation.

Ref: A Domain Specific Supercomputer for Training Deep Neural Networks

Flip any of 23 mantissa bits on Infinity → NaN.

BUT, flipping to NaN is a perhaps lucky coincidence.
What happens when bit flips affect nominal mantissa bits? Exponent bits?

Silent Data Corruption



When 2.0 + 3.0 = 4.0 (0x40800000 instead of 0x40a00000), where's the defect?

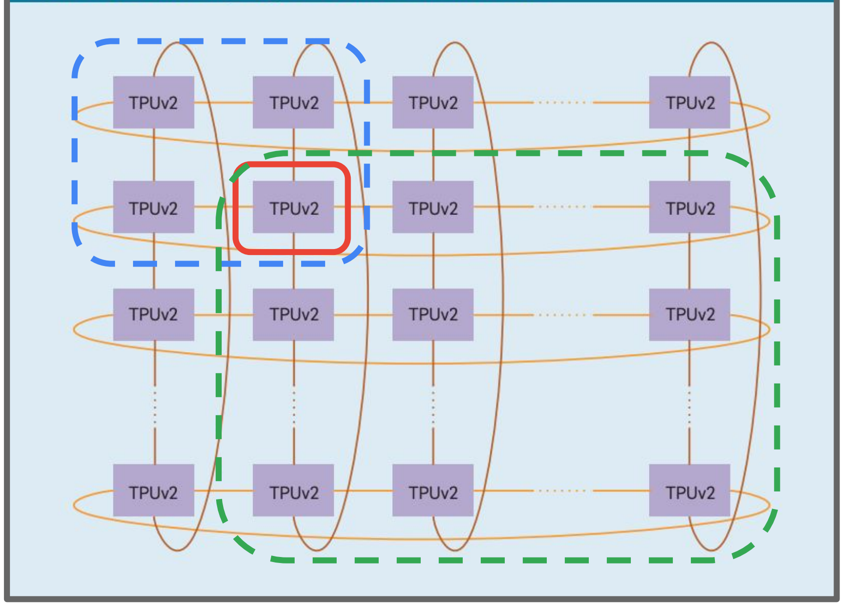
Datacenter SDC Challenges - Peta to Yotta Scale

From the experience of our research application users:

- Extremely hard to identify and debug
  - Large, distributed applications
  - Variability in assigned hardware
  - Failure symptoms are easily confused with other notional ML issues
- Potential for high cost and impact
  - Significant time lost debugging, following the wrong leads, etc.
  - Failure blast radius is potentially large, especially for larger configurations

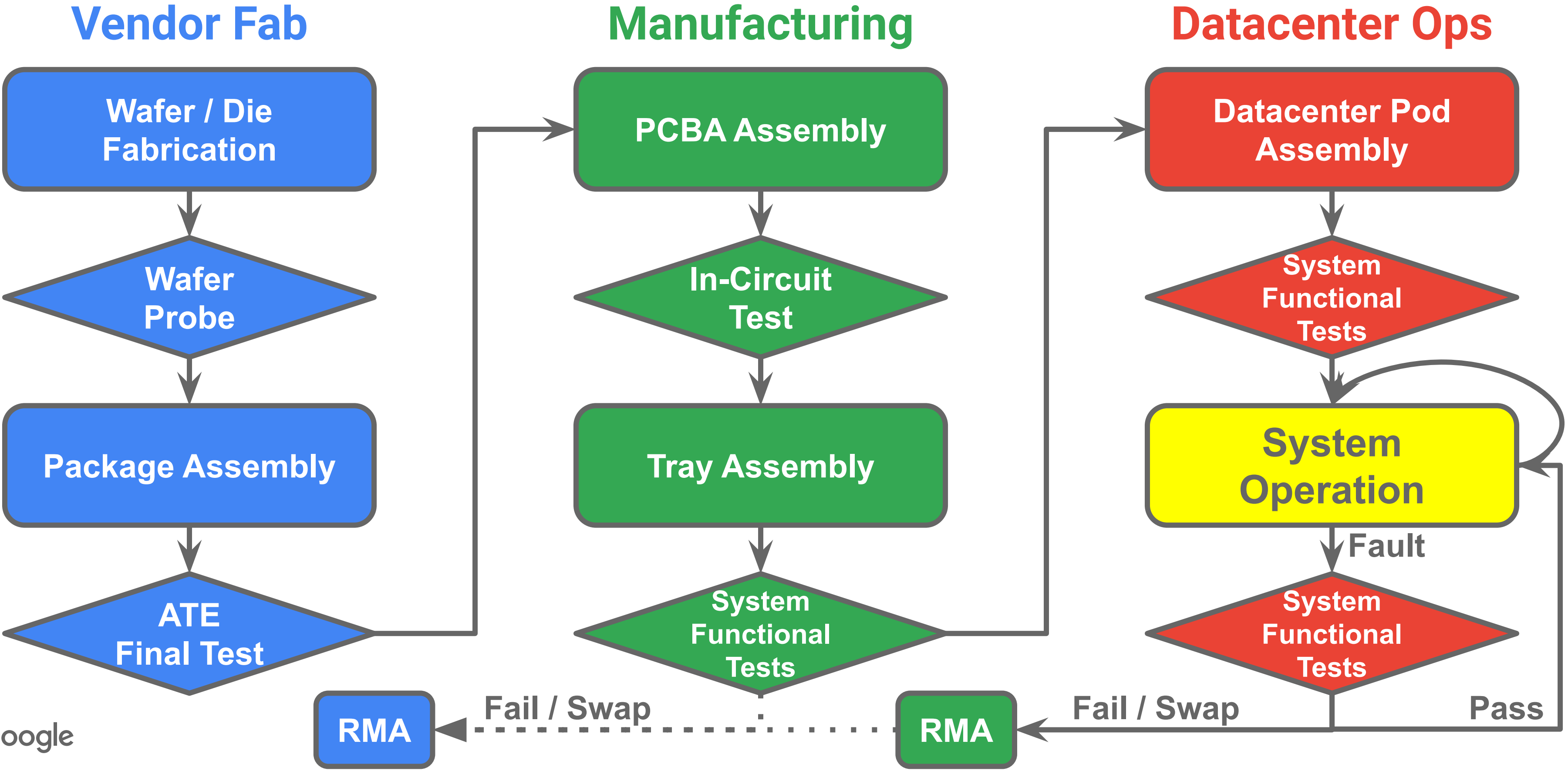
Ref: A Domain Specific Supercomputer for Training Deep Neural Networks

Figure 1. A 2D-torus topology. TPUv2 uses a 16x16 2D torus.



Development & Manufacturing Processes

Fab, Manufacture, Test, and Deploy Processes

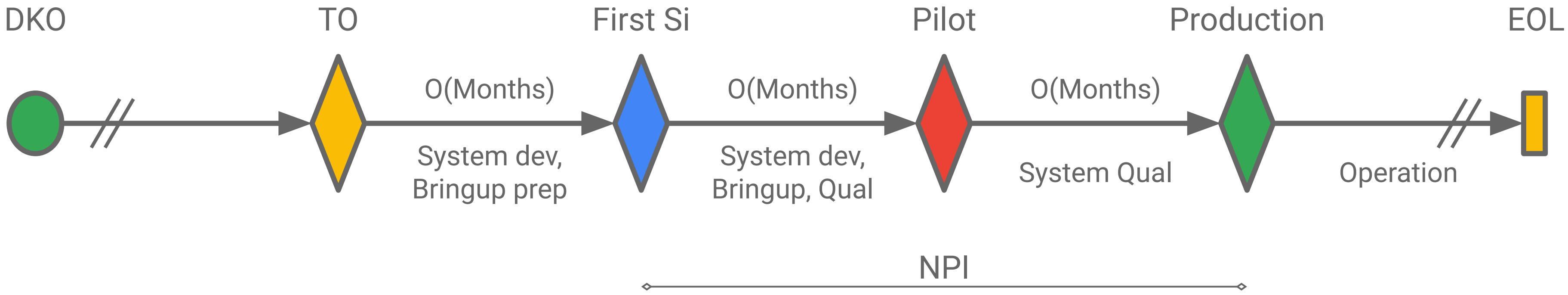


Accelerated Timelines

HW/SW integration, system software, and compiler development all proceed pre-silicon, post tapeout, and throughout NPI.

First silicon to internal full production release can range from 9 - 15+ months.

With short development, long lead times, and deep pipelines of fab-mfg-deploy, There’s precious little time to react!



Fabrication, Manufacturing Test, & Defects

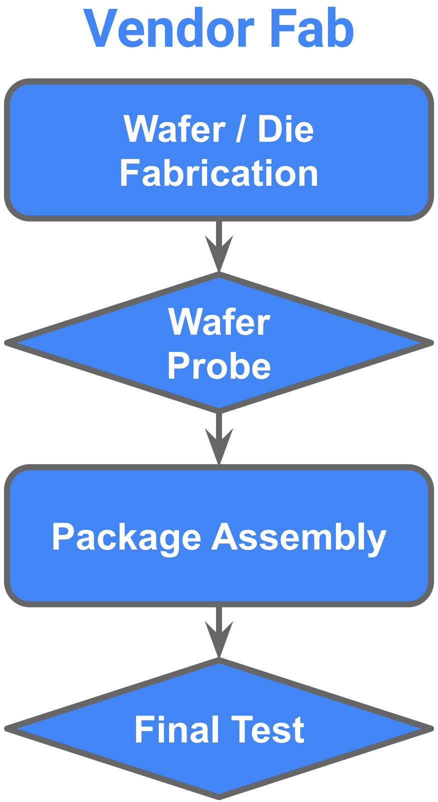
Fabrication ATE Testing

Our vendor-driven Fab testing follows high norms, built on high coverage by DFT structures.

However, NO mission-mode;
Minimal functional testing @ ATE

- ATE Tests
    - >99% Stuck At Coverage
    - >94% Transition Delay Fault Coverage
    - Cell-Aware testing added to address SDC

With billions of transistors, high coverage can still leave millions of untested elements and paths!

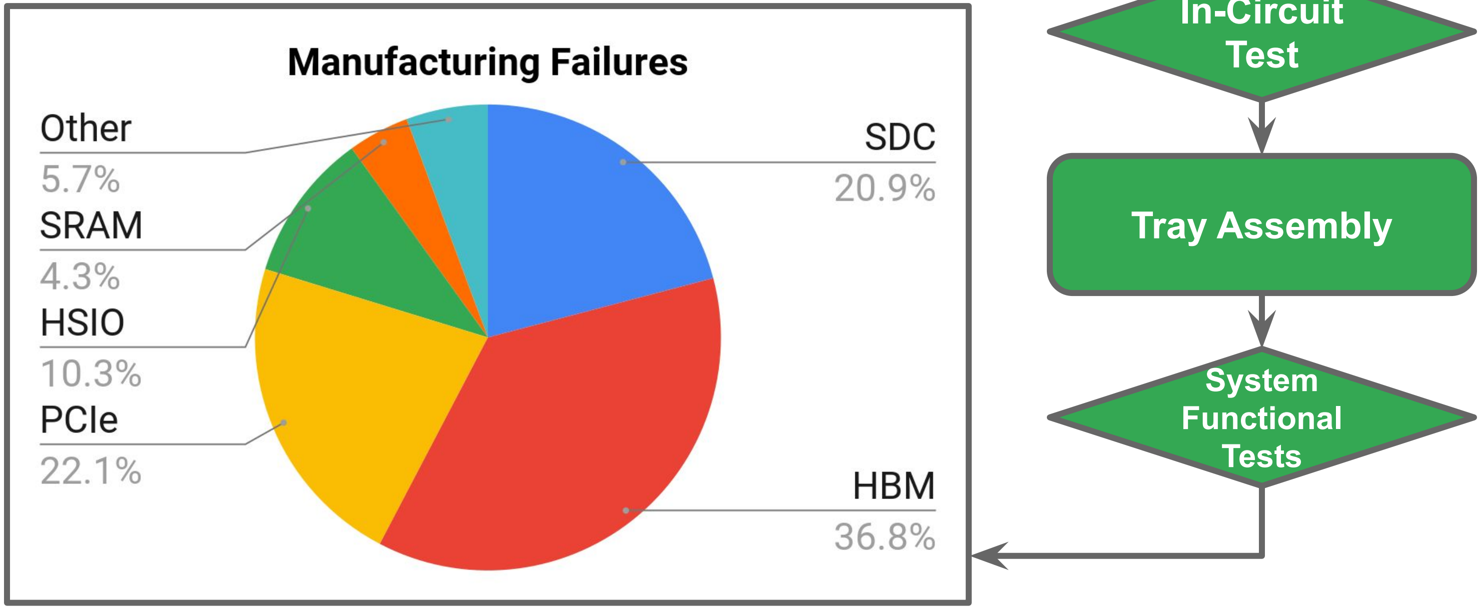


What is the future of DFT, ATE, and Fault Models?

Google

Manufacturing Test

System Functional Test runs full-stack mission mode SW. Utilize BISTs, low-level, directed tests, randomized testing, and full application-level workloads. NO structural testing.



Google

Datacenter Operation

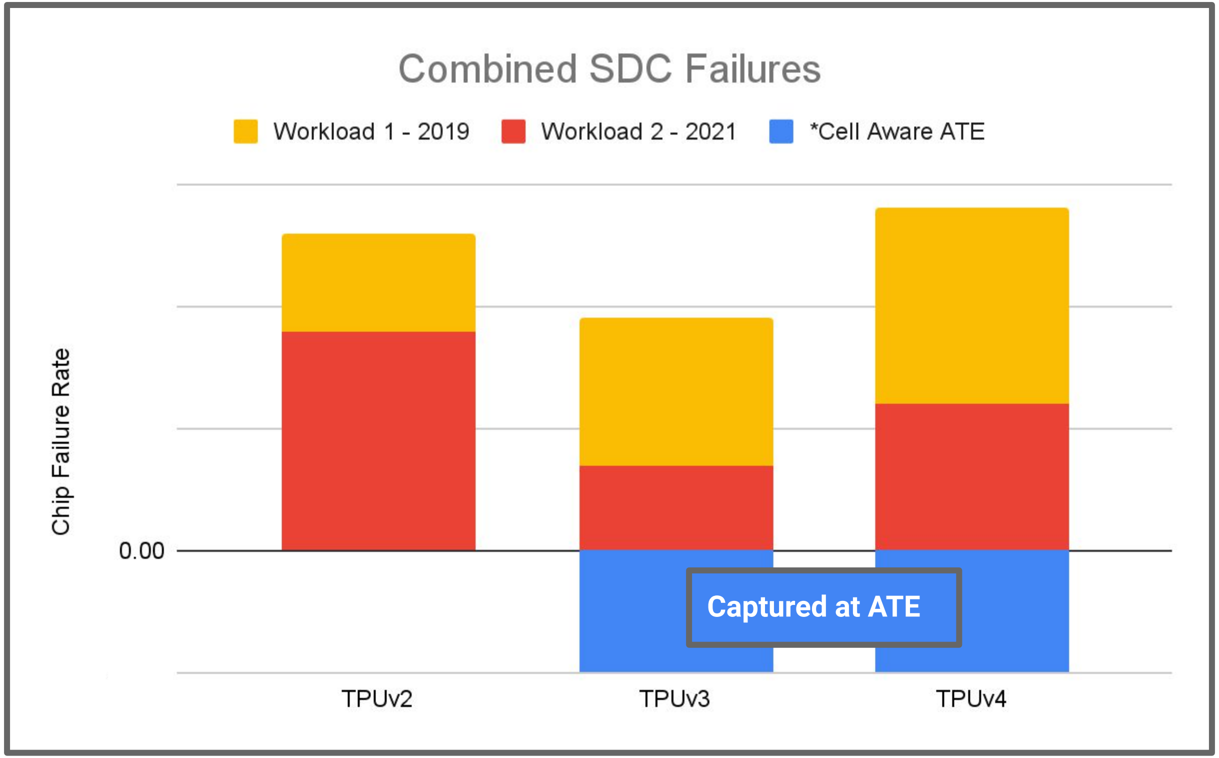
Google

Datacenter Screening Sweeps

Two opportunities to sweep the entire fleet with “newly developed” workloads.

Failing rates fairly consistent across generations.

We’ve been able to move screens to time-0 manufacturing, but this is far from the whole story.



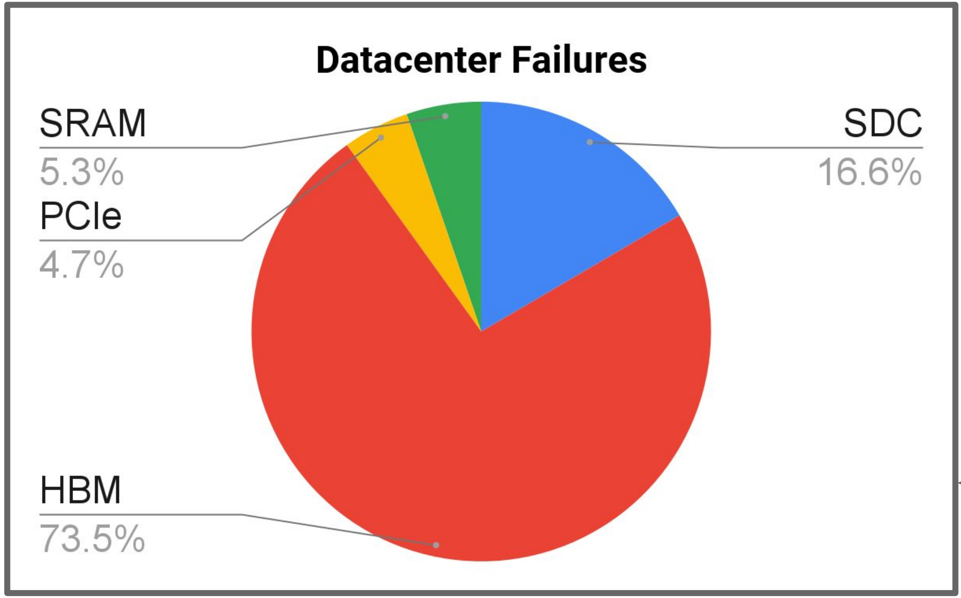
Where will the future land?

Datacenter Operation

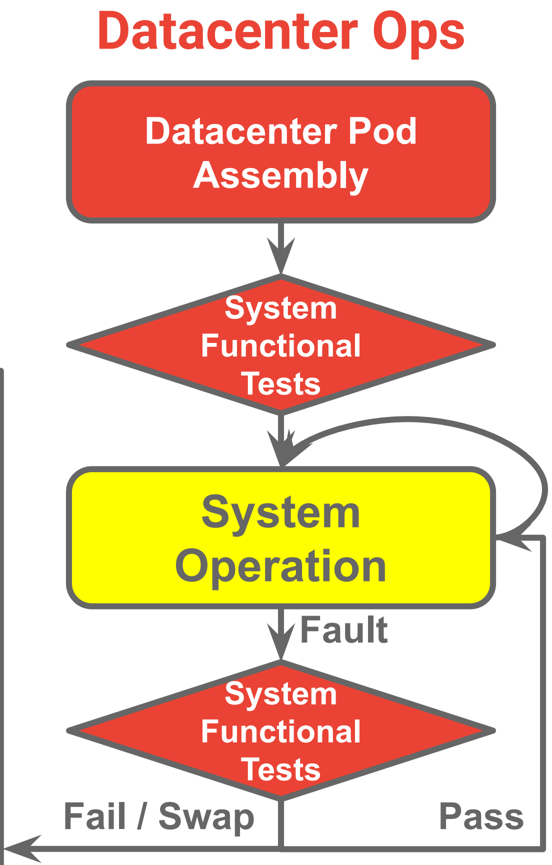
Utilizes nearly the same tests as manufacturing.

We have had a HBM quality issue specifically,
but these are NOISY failures. Obnoxious but manageable.

Total volume VERY similar to mfg.



*Note: Incomplete picture. Data collection & failure confirmation from this stage is very difficult.

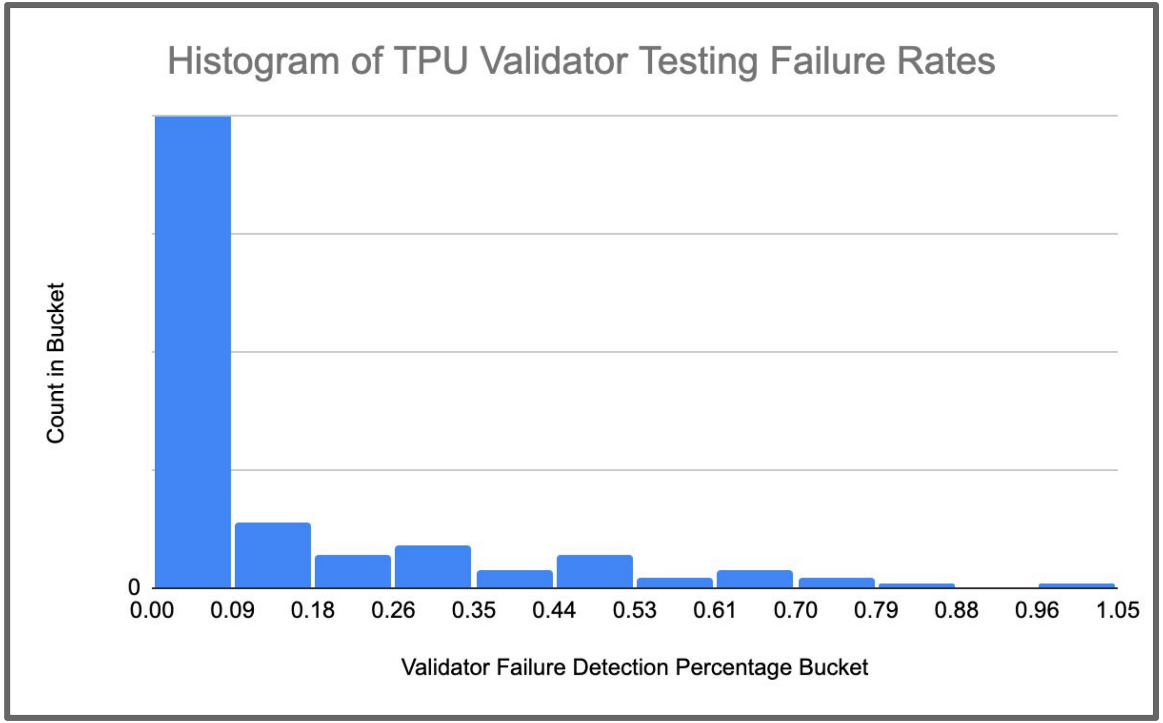


Repeatability Metrics

With background validators in place and automated, repeatability is challenging!

With metrics like these, how do you capture defects in manufacturing?

The majority of machines only fail less than a few times in dozens of testing runs!



When does it stop?

A concern, besides detection and mitigation, is simply-

When will it stop?

How many new workloads are out there that may excite some new pattern of data, instructions, access, whatever - causing that next drip?

What about the affects of time?

This is a call to action for help!

Debug and Defects

Debug and Correlation Efforts - Workload 1 on TPU v3

No correlation between delta-V & Sigma, IDDQ – Indicative of random t-0 defects.

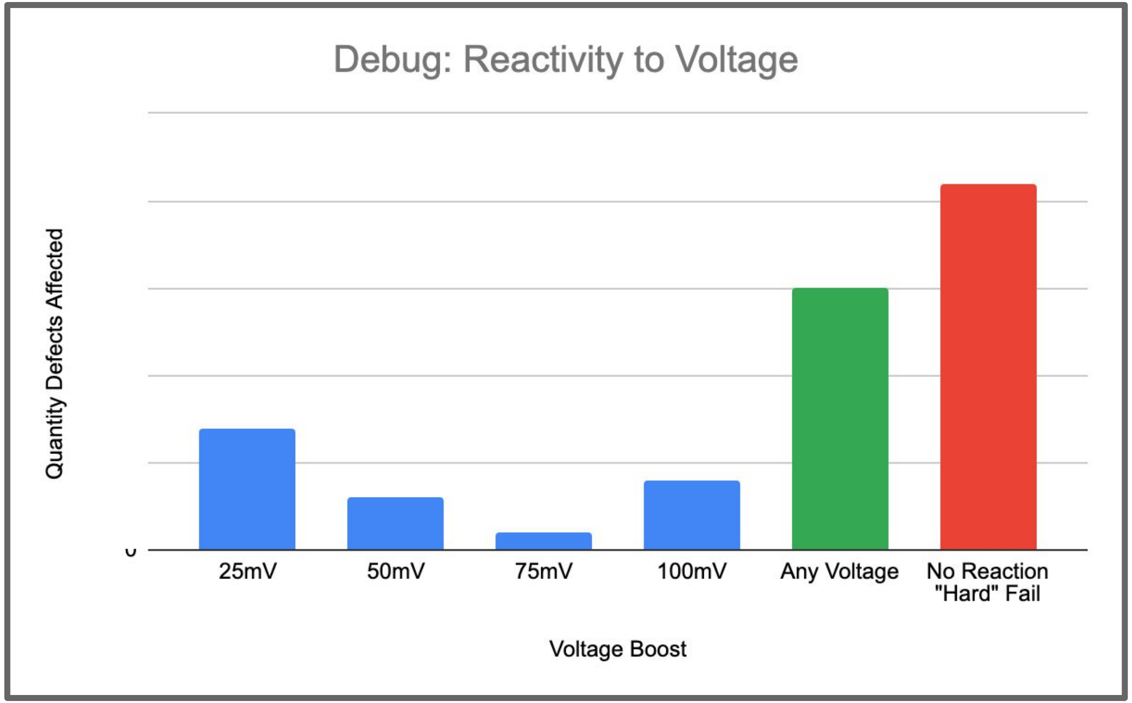
System Functional Tests

Temperatures are stable and reasonable.

No detectable correlation to SRAM ECC correctables.

No correlation to something specific NOR systemic failing location/mechanism has been identified yet.

Efforts are ongoing and increasing!



Google

Random Defect Locales

Defect confirmation with vendor FA indicated random defects.

Defects distributed around computational core and Associated external paths.

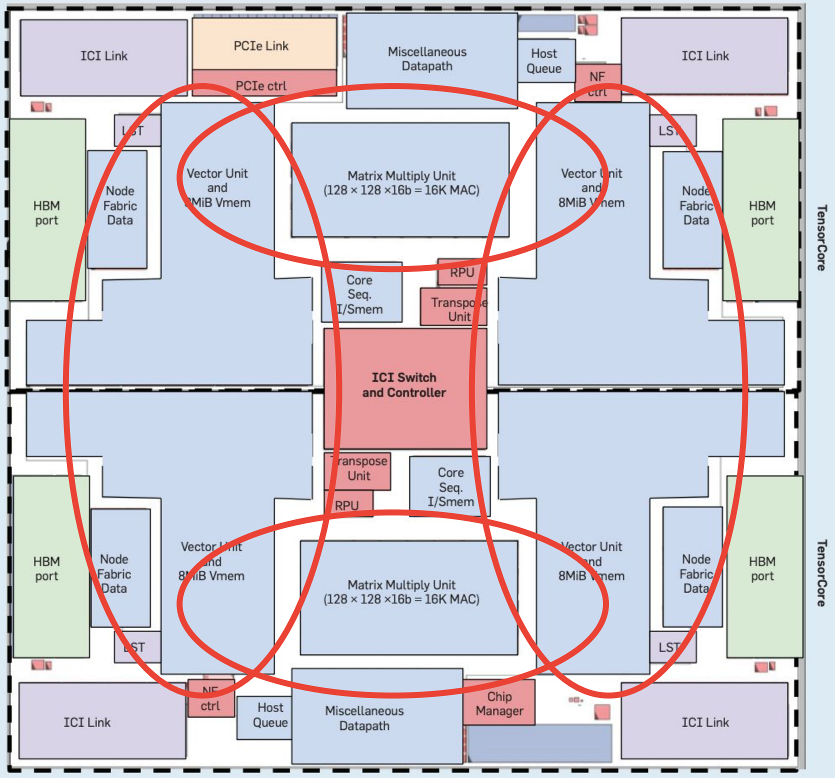
This lead to improved ATE through application of Cell-Aware patterns, But the problem clearly persists.

Final Test
FA

Ref: A Domain Specific Supercomputer for Training Deep Neural Networks

Figure 3. TPUv2 chip floor plan.

It has two TensorCores: Node fabric data and NF controller move on-chip data.



Takeaways & Mitigations

Mitigation Actions - ATE Takeaways

Our ATE and Manufacturing System Functional Tests
seem to have significant overlap gaps. Can, and how, do we close these?

75%+ of SDC ASIC RMAs are NTF @ Vendor,
even with extensive Cell Aware patterns and shmoo testing.

ATE Test

This "traditional" ATE path seems exhausted.

What does the next step-function in quality require?

Considerations & Exploration:

- More ATE / System correlation work
- Process and margins tightening?
- New fault models?
- New ATE testing efficiencies?
- Increased stress or test time?
- Guardband testing

All require investigation and INVESTMENT!

Mitigation Actions - Manufacturing & System Functional Test

- Increase “SLT” - Increase in burnin time & stress
- Increased testing time
  - Defect reduction hinged on statistics?
- Additional guardband testing - Voltage and Frequency
- Partnering with applications to fast-track workload → testcase
- Characterizing existing failures to gather breadcrumbs
- Analyzing workloads; crafting synthetic tests
- Analyzing and grading functional test coverage
- Balancing detection with diagnostics

System Functional Tests

All require investigation and INVESTMENT!

Mitigation Actions - Datacenter Operations

- Constant scanning with validation Daemons
- Preparedness for additional swaps
- Additional tool development to ease debug
- Partnerships for triage
- Processes for defect identification, isolation, and removal

System Tests & Tools

All require investigation and INVESTMENT!

Mitigation Actions - Design and Architecture

- DFx Features - Design for Debug and Design for Test
  - Additional breadcrumbs - Sensors and Monitors
  - BIST engines - Memory, Logic, Functional, what else?
- Detection mechanisms such as parity on unprotected elements
- Mitigation mechanisms - increase in eg. SECDED coverage; Redundancy?
  - Adoption of certain automotive tactics, perhaps?
- Computational protection such as Algorithm-Based Fault Tolerance (ABFT)?
- Space is ripe for research and new innovation

All require investigation and INVESTMENT!

Mitigation Actions - Software Design and Resiliency

- Application resiliency
- Designing with fault tolerance in mind
- Redundancy?
- Self-checking mechanisms?
- Screening development

All require investigation and INVESTMENT!

Final Thoughts and Acknowledgements

Google

Final Thoughts

While investment is needed and won't be free, doing nothing will be far more costly.

A call to action:
We need innovation across the system stack.

- Hardware design and architecture
- Design-for-debug and design-for-test
- ATE techniques and fault models
- Functional System Testing
- Operational screens and mitigations
- Fault Tolerance &
  Resilient application development

We've got our work cut out for us!

Similar issues exist in the CPU space.
See Cores that don't count - Google;
presented in this conference by Peter Hochschild.

Acknowledgements & Thanks

Ahmet Akyildiz

Deepak Asnani

Sandeep Bhatia

Mike Bourland

Ken Durden

Ben Gelb

Rama Govindaraju

Secin Guncavdi

Haitham Hamed

Peter Hochschild

Georgios Konstadinidis

Reiner Pope

Salil Pradhan

Partha Ranganathan

Amir Salek

Boone Severson

Michael Weir

Andy Yang