Testing and Built-In Fault-Tolerance Against Silent Data Corruptions on Computing Chips

Huawei Li

Professor at ICT, Chinese Academy of Sciences

President and Co-founder of CASTEST Inc.

TestConX中国 China

Shanghai ▪ October 31, 2024

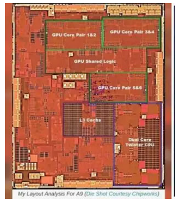
中国科学院计算技术研究所 INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

中科鉴芯 CASTEST

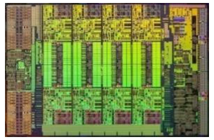
Outline

1) SDCs on Computing Chips
2) Origins of SDCs
3) Testing against SDCs
4) Build-in fault-tolerance (BIFT) against SDCs
   a. BIFT for Multi-Core Systems
   b. BIFT for Deep Learning Systems
5) Concluding Thoughts

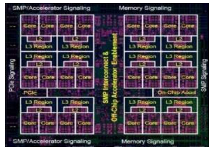
Complexity of typical computing chips



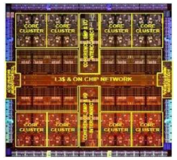
Apple A9
~2B transistors



Intel Haswell-EP Xeon E5
~7B transistors



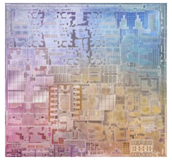
IBM Power9
~8B transistors



Oracle SPARC M7
~10B transistors

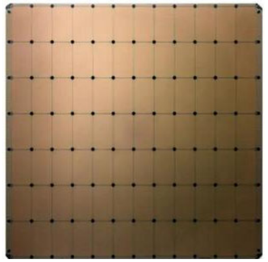
Intel/Altera Stratix 10
~30B transistors

Xilinx VU9P
~ 35B transistors



Apple M1
~ 57B transistors

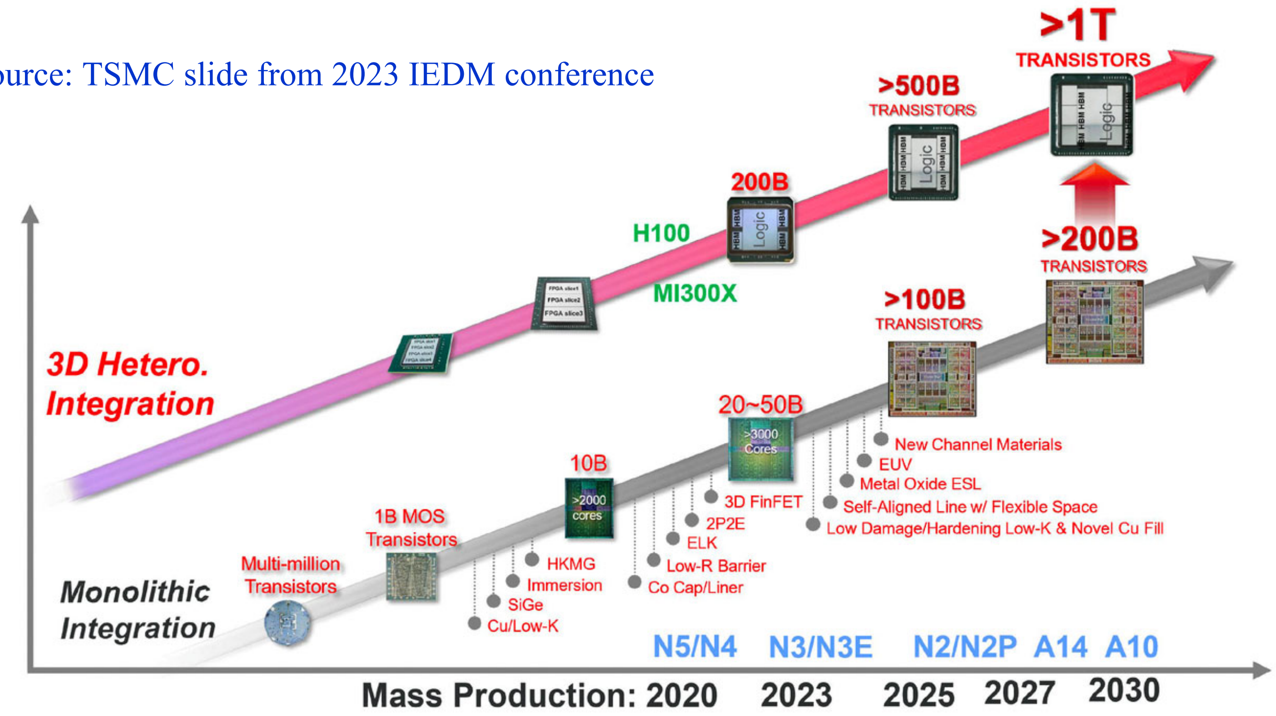
NVIDIA GH100
~80B transistors



Cerebras WSE-3
~ 4T transistors

Trillion-transistor computing chips are coming

Source: TSMC slide from 2023 IEDM conference



Computing chips of increasing complexity are being installed in ever larger numbers in cloud datacenters.

Are they truly reliable for various applications?

- Chips with 200 billion transistors on a single piece of silicon at 1nm-class fabrication processes
- Advancements in packaging: massive multi-chiplet solutions packing more than a trillion transistors

The hidden killer in data centers

- Amazon web services experienced a substantial service outage. (July 2008)
- Facebook lost more than 10% of photos in hard drive failure. (May 2009)
- Google: ephemeral computational errors correlated to components in processors
  - application data corruption and crashes; data corruptions exhibited by various load, store, vector operations; a deterministic AES mis-computation, database index corruption leading to queries being non-deterministically corrupted, ….
- Meta: hundreds of instances of computing errors from processors
  - Spark workloads: core 59 on one processor consistently returned a result of 0 when calculating Int(1.1^{53}), rather than 156. But the same core would return the correct value of 142 for Int(1.1^{52}).

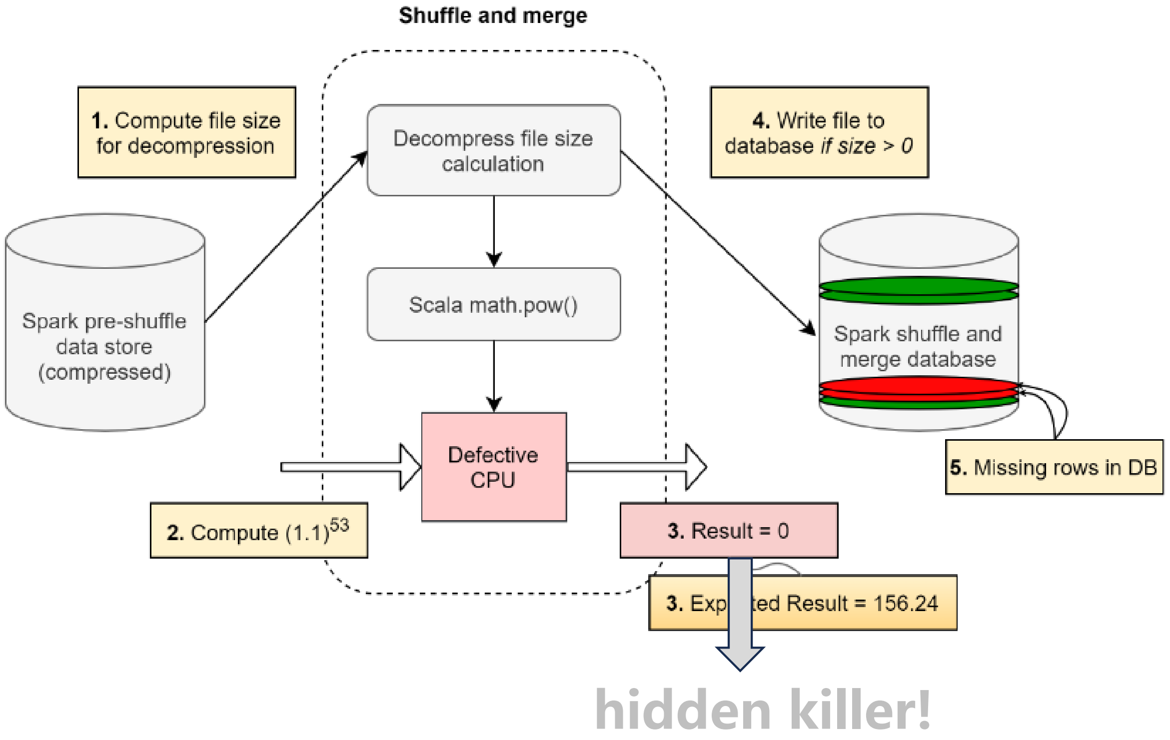
Google: Cores that don’t count,” HotOS ’21, USA https://doi.org/10.1145/3458336.3465297

Meta: Silent Data Corruptions at Scale. arXiv:2102.11245 (2021). https://arxiv.org/abs/2102.11245

The hidden killer in data centers: example

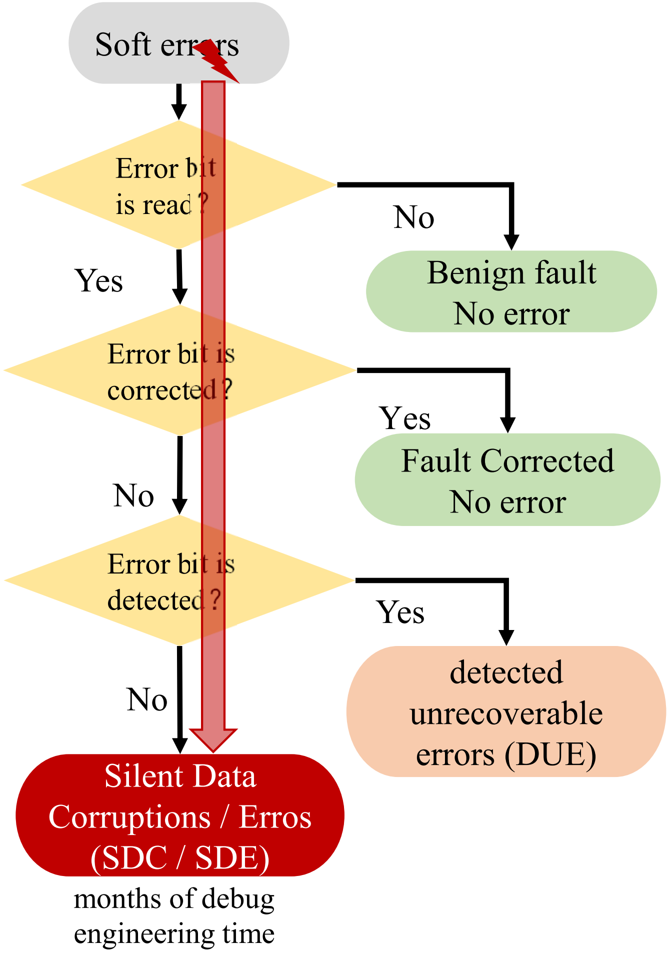
Case that the computing error causes data loss:

- The files are compressed and stored within a data store.
- Before a decompression is performed, the file size is checked to see if the file size is greater than 0.
- A valid compressed file with contents would have a non-zero size.
- When the file size is mistakenly computed as 0, the file was not written into the decompressed output database.



Meta: Silent Data Corruptions at Scale. arXiv preprint arXiv:2102.11245 (2021).

Silent Data Corruptions: the hidden killer in data centers



- SDCs are usually rare events, but in a datacenter running millions of computing chips, 24 hours a day, the rare event becomes an expected occurrence.
- SDCs can have serious impact on largescale infrastructure services when the data corruptions propagate across the stack and manifest as application level problems such as service outage or data loss.
- SDCs can also cause serious impact on safety-critical applications such as autonomous driving.

CPU SDC rate 1/1,000,000 — Meta observed, orders of magnitude higher → 1/1,000 — Addressed widely attention

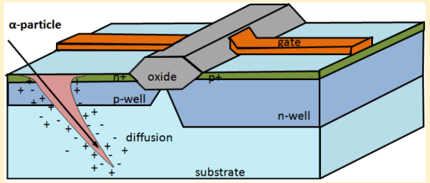
Meta: Silent Data Corruptions at Scale. arXiv preprint arXiv:2102.11245 (2021).

7

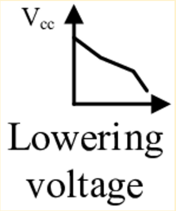
Outline

Origins of SDCs

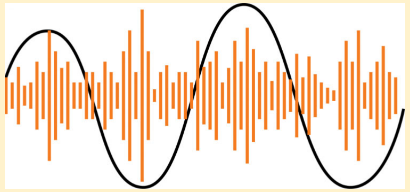
Soft errors: transient, difficult to reappear



Radiation: strike by high-energy particles

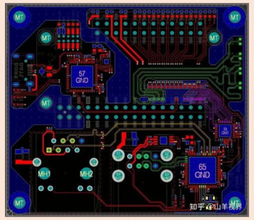


Voltage disturbance

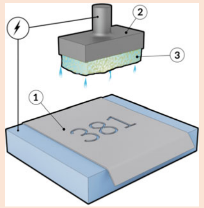


Signal disturbance

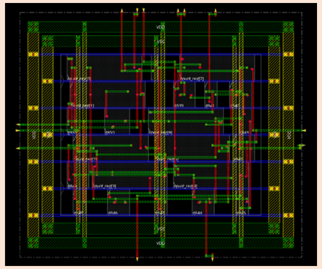
SDC failures: reproducible, not transient



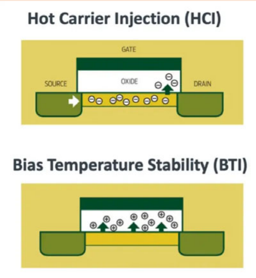
Corner case in design



Unreliable etching



Delay variations



Aging effects

Origins of SDCs

- **Origins of SDCs**
  - **operating conditions, design errors, manufacturing defects & variations, aging**
  - **subtle defects** which create **circuit marginalities** that fail only under the specific combination of temperature, voltage, frequency, and instruction sequence or data set.
- **Fundamental causes of the increasing SDC rate:**
  - ever-smaller feature sizes that push closer to the limits of CMOS scaling,
  - ever-increasing complexity in architectural design (e.g., DVFS),
  - steady increases in CPU scale installed in the system
- New challenges to detect diverse manufacturing defects – especially those defects that manifest in **corner cases**, or only after **post-deployment aging**.

Evaluation of SDCs

- SDC evaluation: *FIT (Failures in Time)*

  1 FIT: one error in a billion (10^9) hours
- The SDC FIT rate of a chip or system:

  the sum of the SDC FIT rates of all its components
- MTTF is inversely related to FIT.

  A FIT rate of 1000 is equivalent to MTTF of ~ 114 years.
- CPU SDCs
  - evaluated within fault injection studies: one in a million (i.e., 1000 FIT)
  - Observed: one in a thousand in datacenters (i.e., MTTF of ~ 41 days)
- Soft-error-rate budgets of IBM Power4 system
  - Chip level: 114 SDC FIT (1000 yr MTTF)
  - System-kill: 4566 DUE FIT (25 yr MTTF)
  - Process-kill: 11415 DUE FIT (10 yr MTTF)

Outline

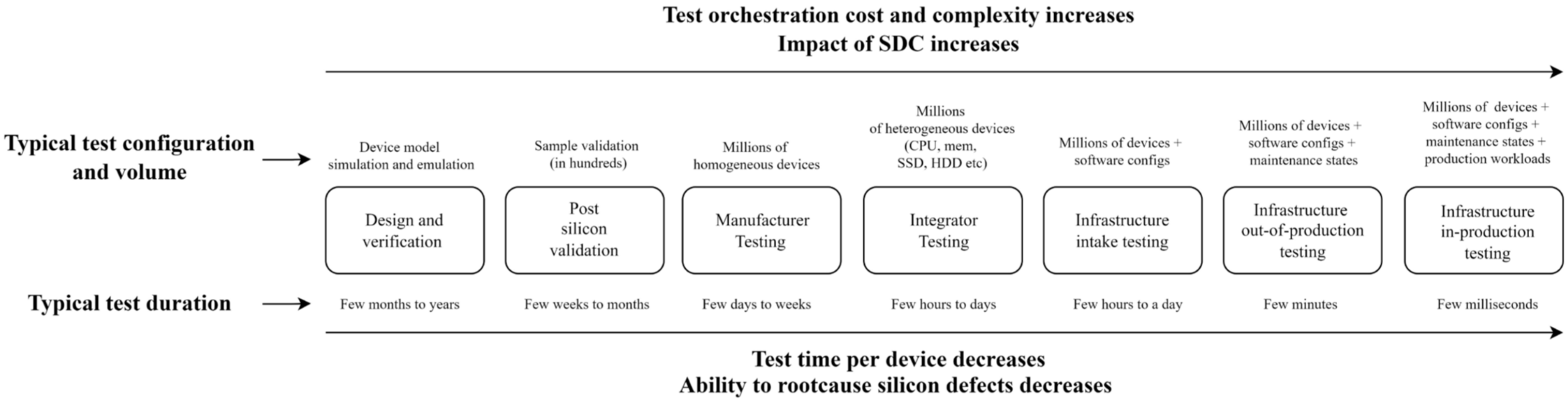
Challenge of Detecting SDCs

- Circuit **marginalities** can result in **random, or unrepeatable, SDCs**.
- Detecting these marginal failures during testing can be extremely difficult because it is impossible to **check every combination of conditions** and **potential workloads**.
- Defects can also be **latent**, meaning they do not show up until after the processors have been running for a long time.

Intel: Data Center Silent Data Errors (2024).

Avoidance of SDCs with various phases of testing

Testing during the whole life-cycle of chips



Meta: Detecting silent data corruptions in the wild, arXiv:2203.08989[2022]

Testing against SDCs: design verification

Test orchestration cost and complexity increases
Impact of SDC increases

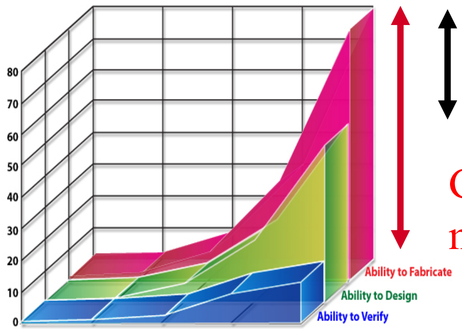
Typical test configuration and volume →

Device model simulation and emulation	Sample validation (in hundreds)	Millions of homogeneous devices	Millions of heterogeneous devices (CPU, mem, SSD, HDD etc)	Millions of devices + software configs	Millions of devices + software configs + maintenance states	Millions of devices + software configs + maintenance states + production workloads
Design and verification	Post silicon validation	Manufacturer Testing	Integrator Testing	Infrastructure intake testing	Infrastructure out-of-production testing	Infrastructure in-production testing
Few months to years	Few weeks to months	Few days to weeks	Few hours to days	Few hours to a day	Few minutes	Few milliseconds

Typical test duration →

Test time per device decreases
Ability to rootcause silicon defects decreases

Design verification: The complexity is much higher than design itself.

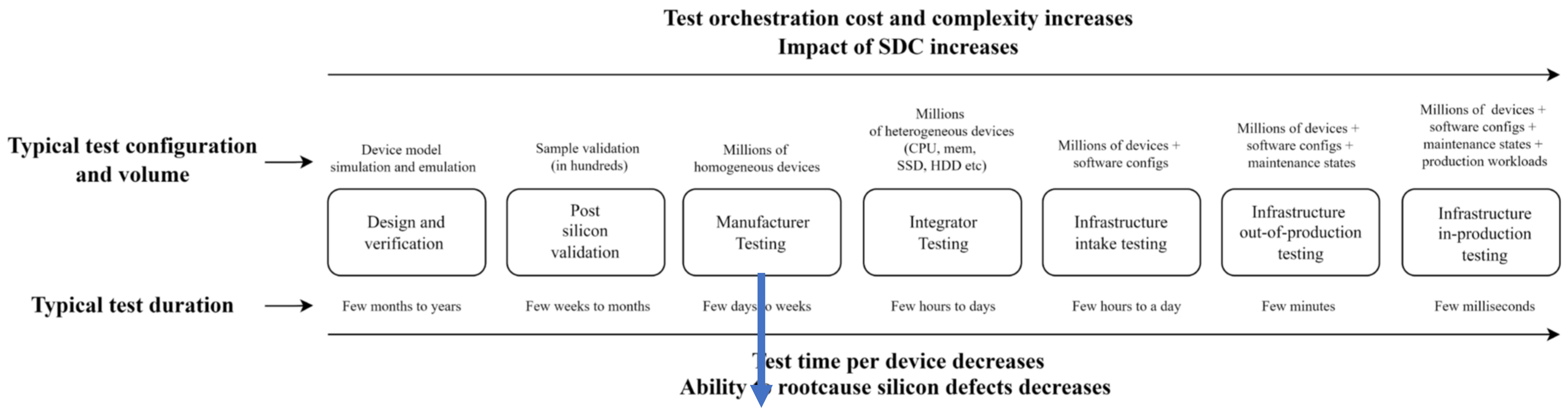


Gap of Design

Gap of Verification: missing verification on corner cases

Source: ITRS

Testing against SDCs: Manufacture Testing



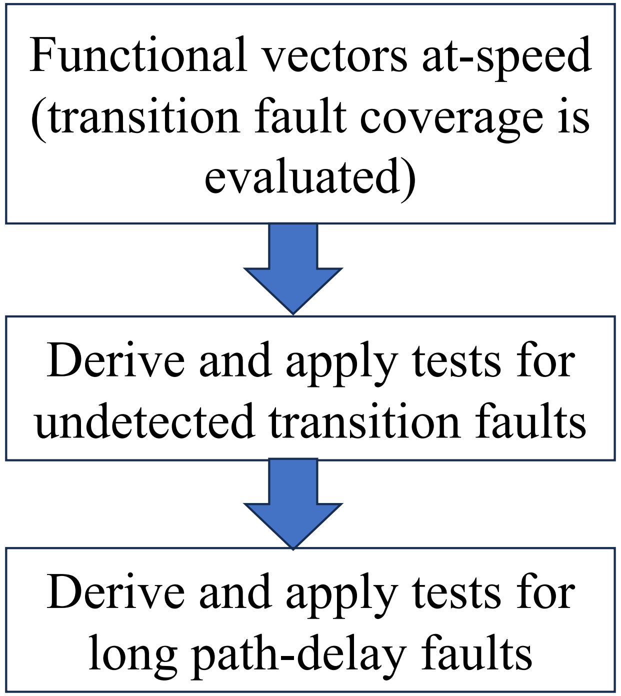
Manufacture Testing:
With the increasing variations (affecting delay), the defect coverage of testing cannot meet request of high reliability.

- Process Variations
- Crosstalk Noises
- Small delay defects
- Power Noises

16

Manufacturer testing considering delay variations

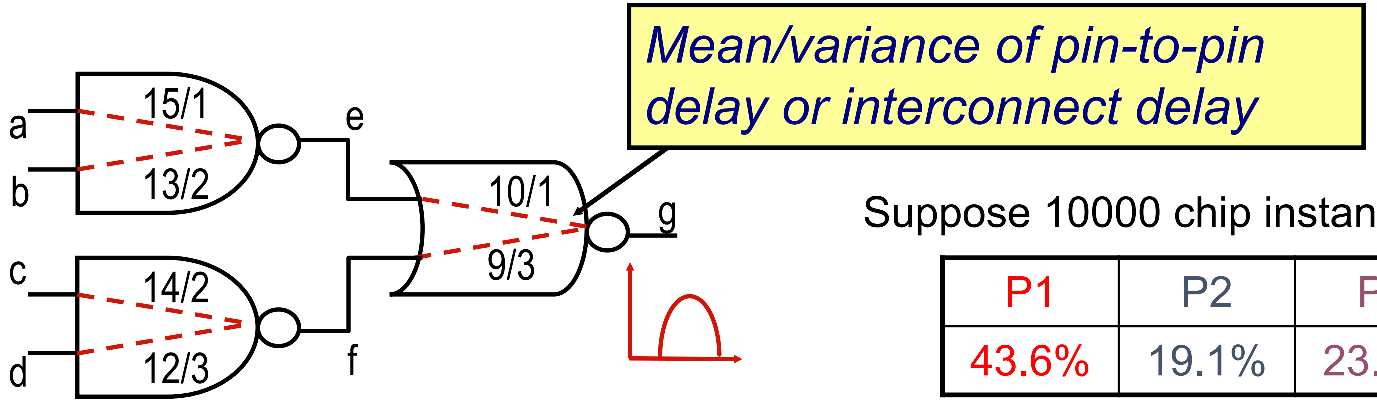
A cost-effective at-speed test flow



- With nano-meter technologies, conventional transition and path delay fault models and at-speed test methodologies are severely challenged!
- No. of critical paths increases due to speed and power saving techniques.
- Delay variability increases due to process, defect, temperature, power, and noise factors: **small delay faults**

SDF testing using SSTA for statistical long paths

- Statistical static timing analysis (SSTA)
  - Circuit delays can be modeled as correlated random variables to take various local & global factors into account
- Delay testing should target statistical long paths whose tests maximize the detection of small delay faults



P1: a, e, g
P2: b, e, g
P3: c, f, g
P4: d, f, g

Suppose 10000 chip instances are produced:

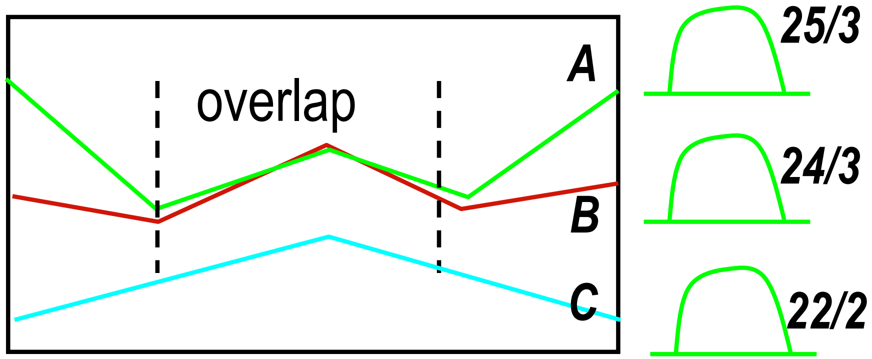
P1	P2	P3	P4
43.6%	19.1%	23.6%	13.7%

L.-C. Wang, et al., "Critical path selection for delay fault testing based upon a statistical timing model," TCAD 2004.

SDF testing considering path correlation

- Considering the path correlation for path selection
  - Achieves higher test quality with the same number of selected paths
  - Selects fewer paths to achieve same level of test quality
  - Monte Carlo simulation can be used, but time-consuming

Output arrival times: mean/standard deviation



After selecting path A, should path B or C be selected?

L.-C. Wang, et al., "Critical path selection for delay fault testing based upon a statistical timing model," TCAD 2004.

19

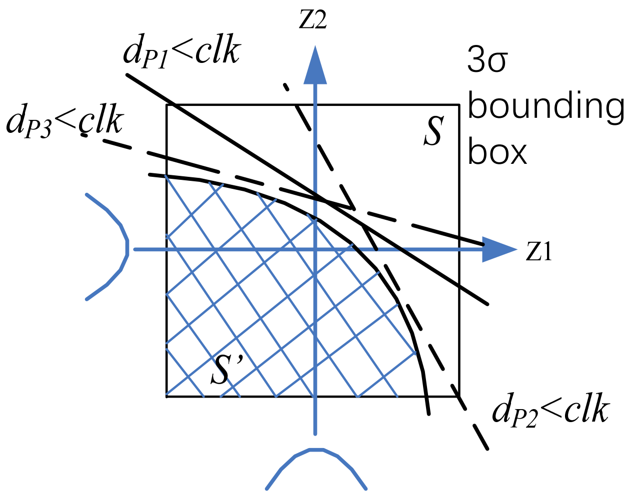
SDF testing with SSTA and path correlation

- Statistical static timing model for gate/wire

$$d_a = \mu_a + \sum_i^n a_i z_i + a_{n+1} R$$

z_i, R- random variables (RV's) modeling spatial correlated variation

- Circuit delay: $(n+1)$-*dimensional space* S: Cartesian product of all RV's



S_P: where path P meets delay constraint $(d_p < clk)$

S': where the circuit meets delay constraint $(d_{circuit} < clk)$

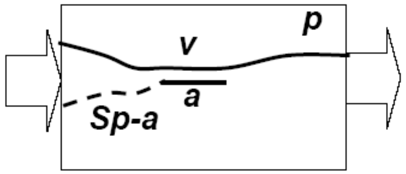
$$S' = \bigcap_{\text{for each path } P} S_P \qquad S_H = \bigcap_{Pi \text{ in } H} S_{Pi}$$

Path selection: Given number of paths to be selected, **finding a path set H with the minimal S_H** instead of with several smallest S_{pi} (longest paths)

Zijian He, Tao Lv, Huawei Li, Xiaowei Li: Test Path Selection for Capturing Delay Failures Under Statistical Timing Model. IEEE Trans. VLSI Systems (TVLSI), 2013

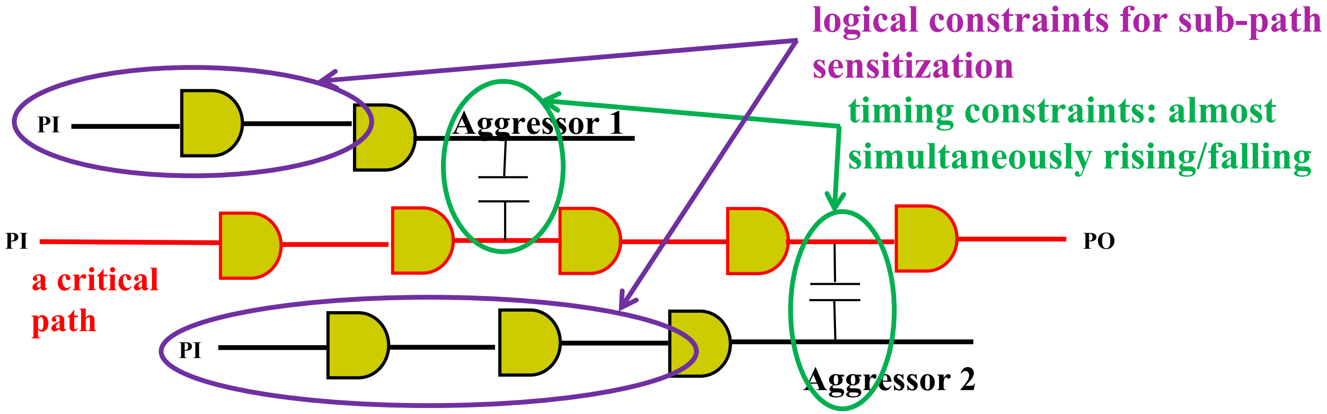
SDF testing considering crosstalk/power noises

- Identifying a test that corresponds to the longest delay along the target path
  - Path delay is highly pattern dependent
- Activating the worst-case crosstalk/power noises during test generation
  - Precise crosstalk-induced path delay fault (PCPDF)
  - Fault collection considering coupling capacitance and timing after place & routing



$(p, \{sp\text{-}a_i, \langle v_i, a_i \rangle\}), i=1,2,\ldots,n$

PCPDF model



Minjin Zhang, et al.: Path Delay Test Generation Toward Activation of Worst Case Coupling Effects (TVLSI 2011)

Xiang Fu, et al.: Robust Test Generation for Power Supply Noise induced Path Delay Faults (ASP-DAC 2008)

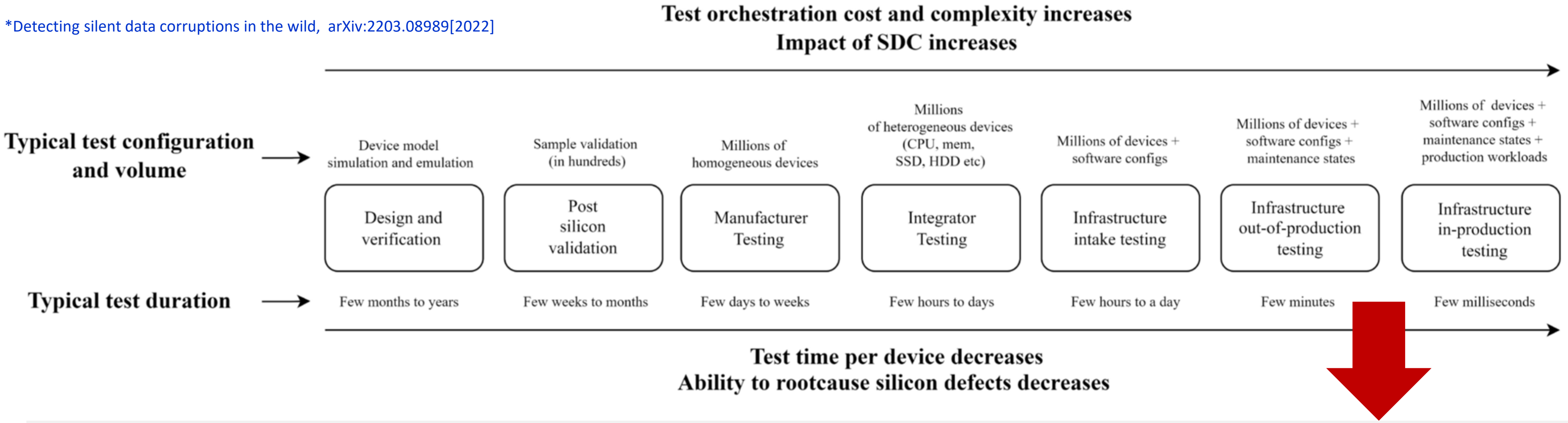
Manufacturer testing against SDCs

- Other suggestions for improving test quality
  - Bridging tests: enumerate likely bridging fault sites (interconnects) by layout simulation
  - Cell-aware Test: effective to FinFET technology
  - IDDQ tests: test by measuring current flow
  - TARO (transition fault propagation to all reachable outputs): for each given transition fault, generate tests for each reachable output
  - N-detect stuck-at / transition tests: detect every stuck-at / transition fault n times by targeting different sensitive paths
    - The pattern count increases approximately linearly with n
    - Effective test selection to choose a small test set with high test quality

Dawen Xu, et al.: Test-Quality Optimization for Variable n-Detections of Transition Faults Prediction (TVLSI 2014)

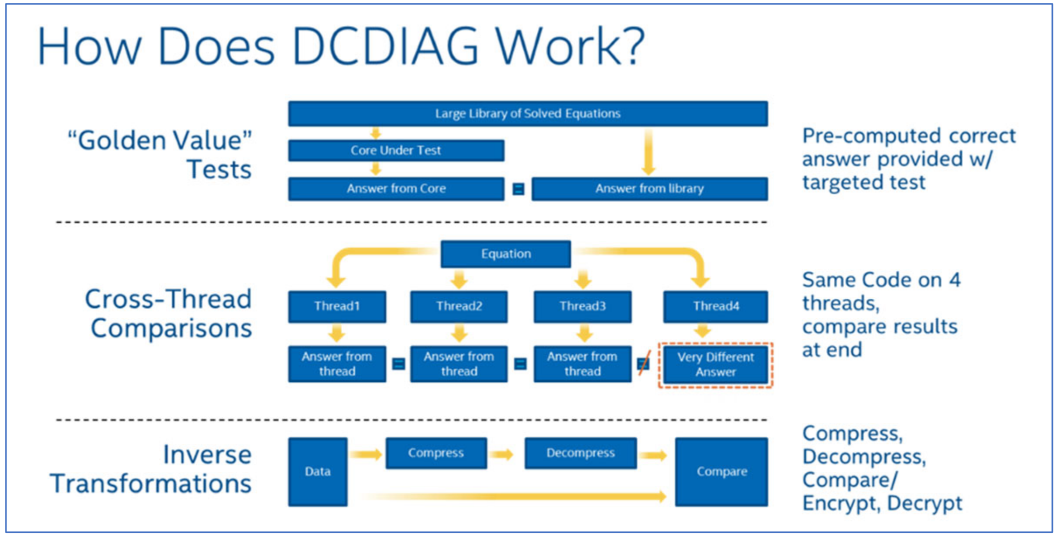
Testing against SDCs: at System-Level

*Detecting silent data corruptions in the wild, arXiv:2203.08989[2022]



- Error checking schemes
  - data paths and memory storage may include parity or error check and ECC schemes
  - network packets have error-detecting or error-correcting algorithms such as CRC
- Periodic system-level tests using multiple operating system environments
  - Intel Data Center Diagnostics Tool (DCDiag) to identify faulty processors

System-level tests in DCDiag (Intel)



- **Golden value tests**: e.g. the square root of 2 or the SHA-1 checksum of a fixed input, with expected value
- **Cross-thread comparisons**: all cores run the same sequence of instructions using the same dataset, while the output generated by each core is compared against that produced by other cores.
  - Variability: randomly generated datasets and/or randomizing the order or selection of processor instructions
- **Inverse transformation tests**: execute two operations back-to-back to arrive at the original input, e.g.
  - compression and decompression, encryption and decryption, on randomly generated dataset
- Others: non-compute functions, e.g., core-to-core / socket-to-socket communications, caches, interrupts, …

https://www.intel.com/

24

Testing against SDCs: Summary of system-level tests

- Tests conducted **periodically in the system environment**
- Test programs of SLT
  - check **every instruction** on each core, all the caches, core-to-core communications, memory interfaces, uncore functions, with the purpose of exercising a **high percentage of transistors**.
  - rely on **pseudo-random data** and **combinations of instructions**
  - repeated looping for **millions of clock cycles** to span the vast data, address, and instruction space
- At a rate of 10 failures in time (FIT) for each chip, a data center of modest size (100,000 chips) is likely to experience at least one SDC event every month.
- To minimize the rate of SDCs, periodic testing of data center infrastructure to identify defective components is a critical aspect of maintenance.

Intel: Optimization of Tests for Managing Silicon Defects in Data Centers https://ieeexplore.ieee.org/document/9983919

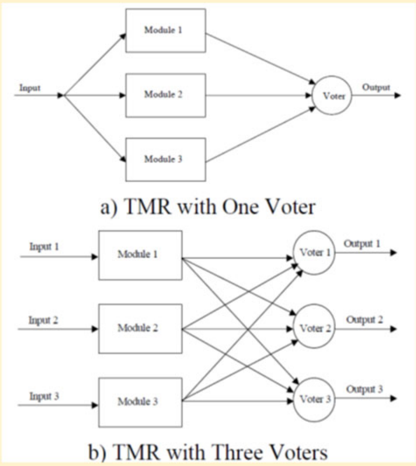
Outline

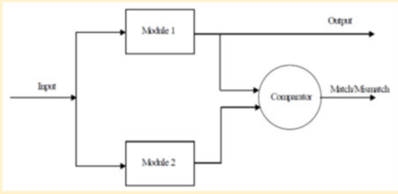
1) SDCs on Computing Chips
2) Origins of SDCs
3) Testing against SDCs
4) Build-in fault-tolerance (BIFT) against SDCs
    a. BIFT for Multi-Core Systems
    b. BIFT for Deep Learning Systems
5) Concluding Thoughts

Classical fault tolerance: error detection & recovery

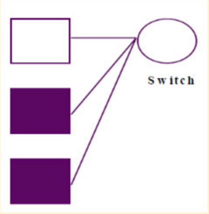
Space Redundancy

TMR





DMR



Standby

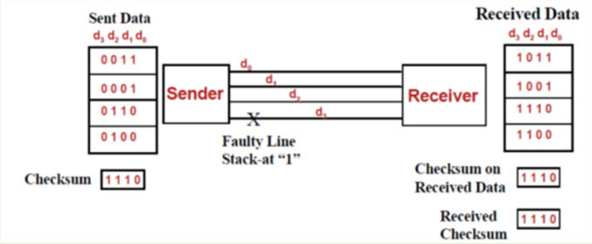
Information Redundancy

Parity

c_1	c_2	c_3	c_4	
0	0	0	0	No errors
x_1	x_2	x_3	1	Single error (in a position $x_1x_2x_3$) is detected and can be corrected
y_3	y_2	y_1	0	Double error is detected but cannot be corrected
0	0	0	1	Error in parity bit p_4

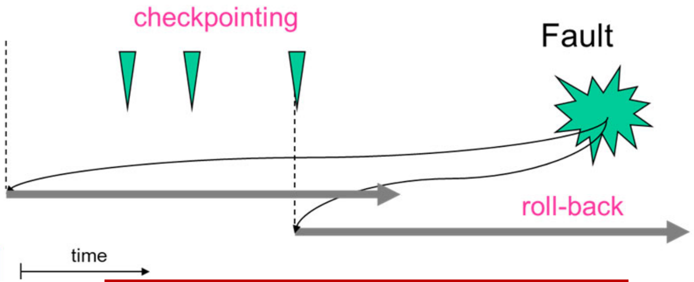
SECDED ECC

Computation: Arithmetic Coding



communication: Checksum

Time Redundancy



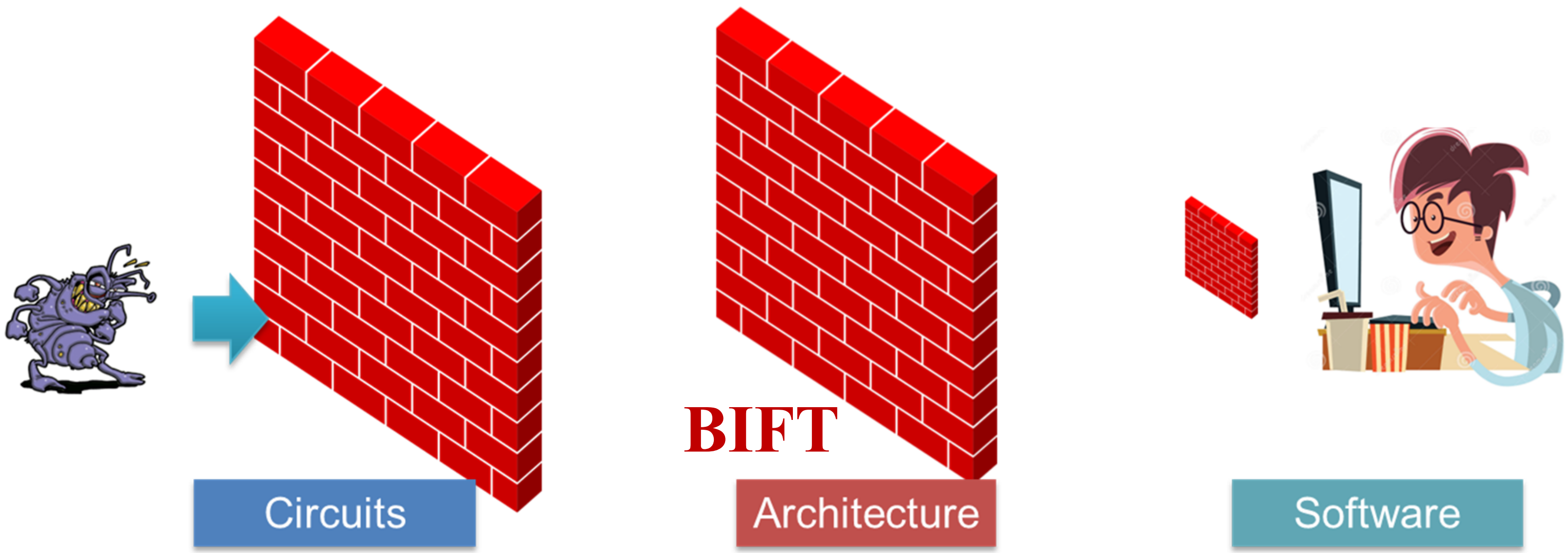
Recompute & Compare

Checkpoint & Rollback

27

From Error Detection to Recovery - BIFT

Self Test - Self Diagnosis - Self Repair（3S）
Defenders: Build-in Circuits, Architecture, or software



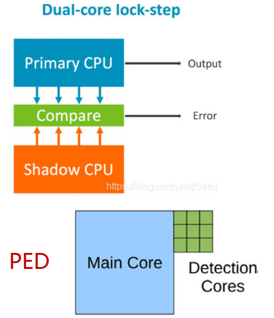
28

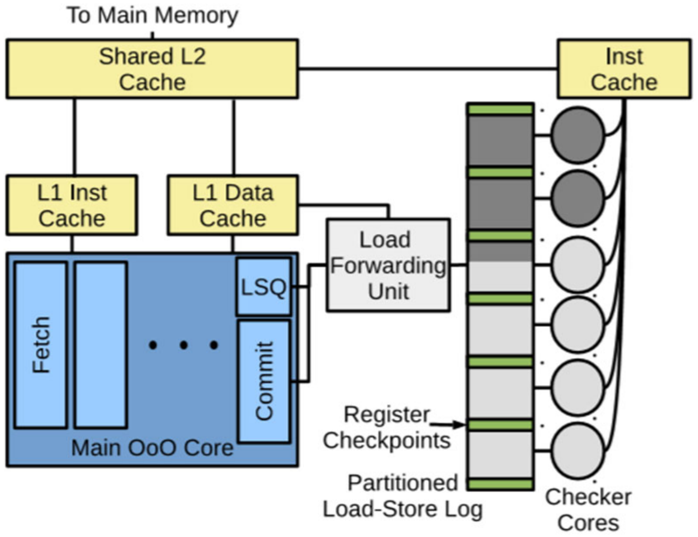
Outline

1) SDCs on Computing Chips
2) Origins of SDCs
3) Testing against SDCs
4) Build-in fault-tolerance (BIFT) against SDCs
    a. BIFT for Multi-Core Systems
    b. BIFT for Deep Learning Systems
5) Concluding Thoughts

a. BIFT for Multi-core Automotive MCUs

- Lockstep offers a high error coverage at the cost of >100% area & power consumption overhead.
- Parallel Error Detection (PED) Using Heterogeneous Cores [DSN-18]
  - several lower-performance cores to run the program segments of the main core



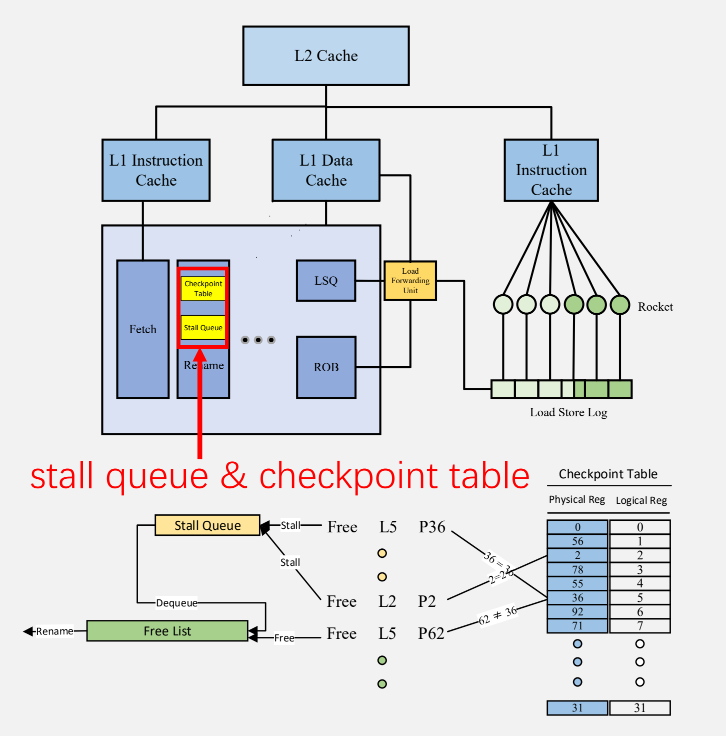


Problems:

- reduces the main core's performance
  - Use the beginning and end of the program segment as checkpoints
  - At each checkpoint, it requires the main core to suspend committing instructions for several clock cycles, and copy the register states to the check core
- causes high error detection latency
  - The performance of check cores is much lower than the main core, => much longer runtime in comparison with the main core
  - Resulting high error detection latency (i.e., 15,000 cycles).

PED with Low Performance impact to the main core

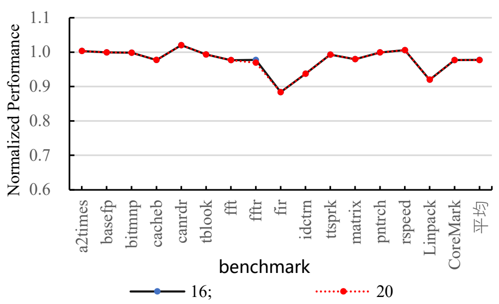
- Lockstep offers a high error coverage at the cost of >100% area & power consumption overhead.
- PED Using Heterogeneous Cores (several lower-performance cores to run the segments of the main core), reduces the main core's performance, and causes high error detection latency (i.e., 15,000 cycles).



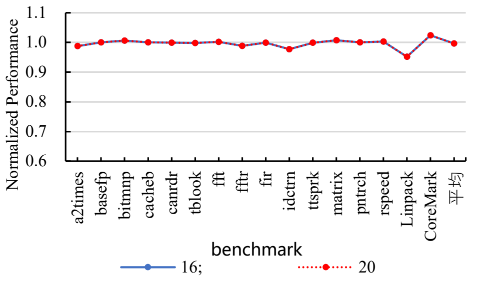
checkpoint state copy

- stalls the release of physical registers corresponding to checkpoint states
- No needs to stop instruction commission while copying register states, thus reduces the impact of error detection on the main core's performance
- The performance impact to the main core: under 1%

Previous: suspend committing instructions

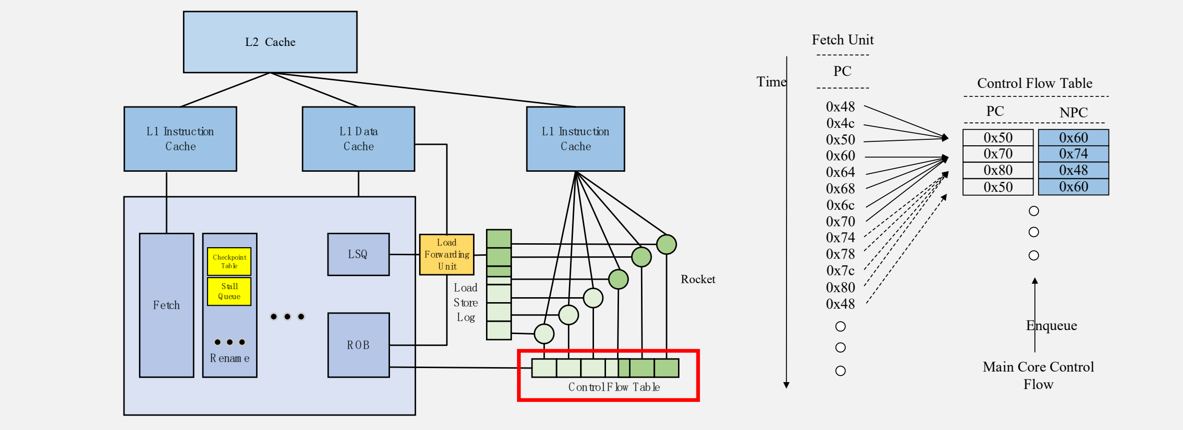


New: stall the release of physical registers



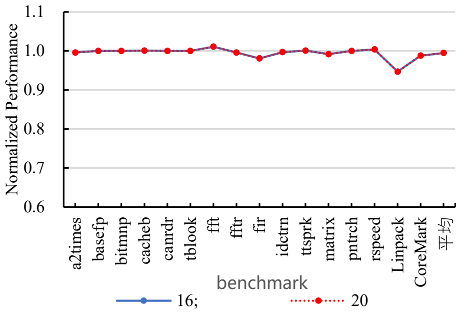
PED with Low error detection Latency

- Lockstep offers a high error coverage at the cost of >100% area & power consumption overhead.
- PED Using Heterogeneous Cores (several lower-performance cores to run the segments of the main core), reduces the main core's performance, and causes high error detection latency (i.e., 15,000 cycles).

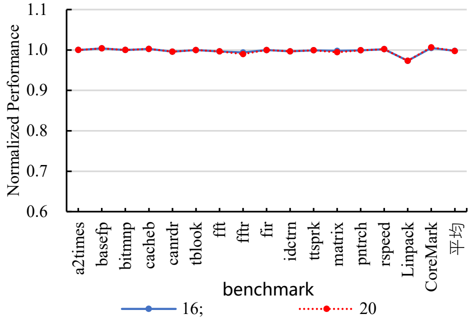


- The main core's control flow is used to guide the check core's instruction fetch, ensuring correct fetching each time and eliminating the overhead of check core branch prediction failures.
- The performance of the check core improves by an average of 15%.
- The error detection latency is controlled within 2,000 cycles, far less than the previous method's 15,000 cycles.

Previous: branch prediction by the check core itself

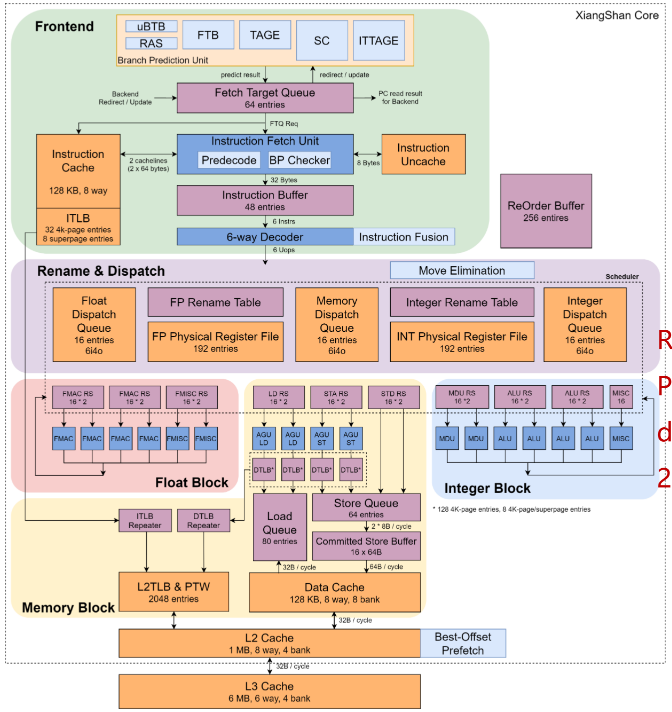


New: branch prediction guided by the main core's control flow



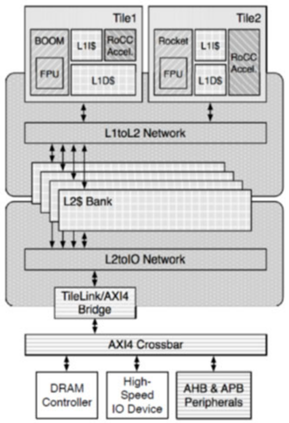
Implementation of Low Latency PED on RISC-V processors

main core



Ratio of Performance difference 20 : 1

with 12/16 check cores



Xiangshan: a high-performance processor

Rocket: a low- performance processor

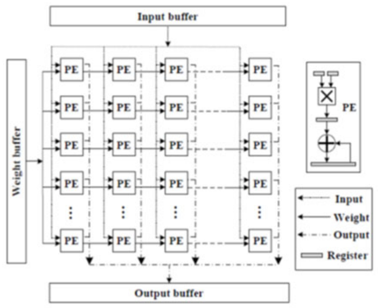
- Baseline: Lockstep with more than 100% area and power overhead.
- When using 12 check cores, the performance overhead is 1% on average, the logic area overhead is 38%, the memory area overhead is 17%, while the power overhead is 16.4%.
- When using 16 check cores, the performance overhead is 0.1% on average, the logic area overhead is 50%, the memory area overhead is 22%, while the power overhead is 21.8%.
- The error detection latency is controlled within 2,000 cycles.

Zhefan Lv, et al.: Heterogeneous Architecturally Parallel Error Detection with Low Error Detection Latency for Highly Reliable Automotive Electronic Systems (JCAD 2023)

33

Outline

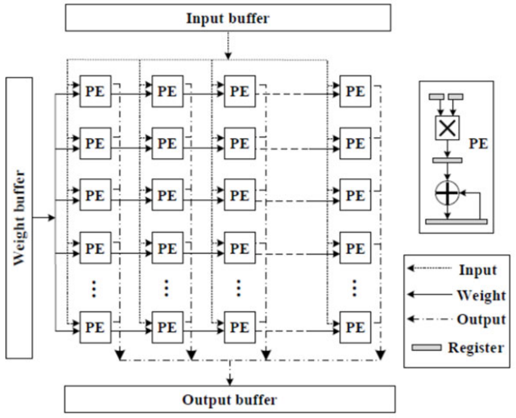
1) SDCs on Computing Chips
2) Origins of SDCs
3) Testing against SDCs
4) Build-in fault-tolerance (BIFT) against SDCs
   a. BIFT for Multi-Core Systems
   b. BIFT for Deep Learning Systems
5) Concluding Thoughts



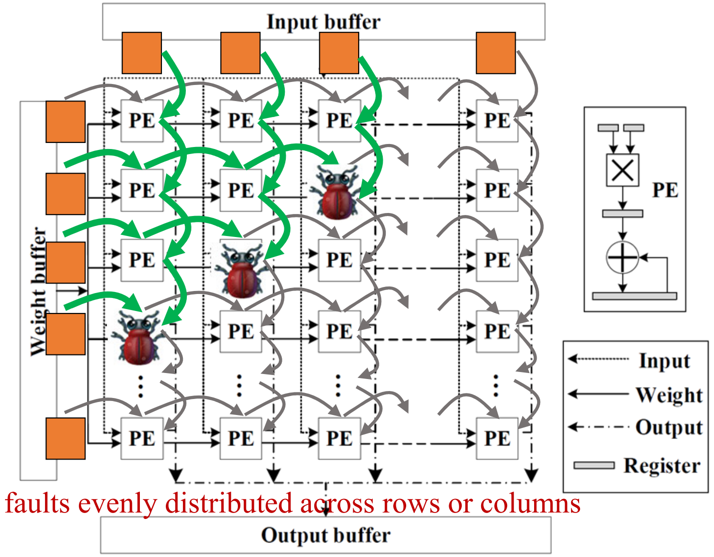
2-D computing array of a typical DLA

b. BIFT for Deep Learning Systems

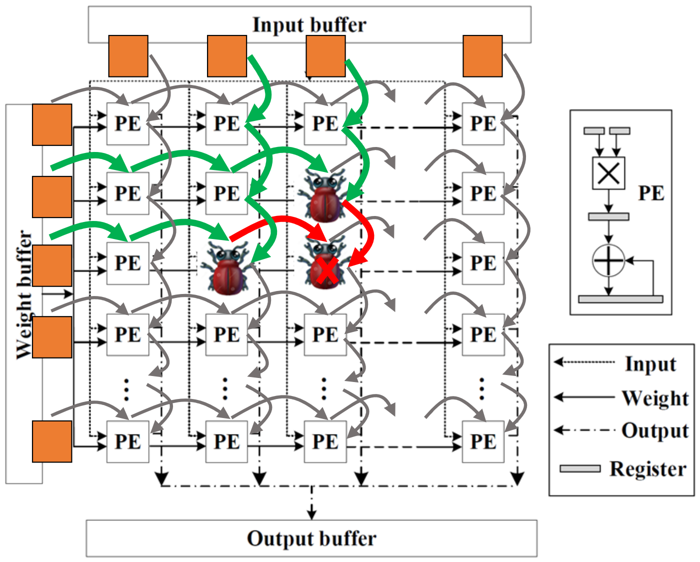
Traditional Array Redundancy Strategy



A typical DLA with 2D computing array



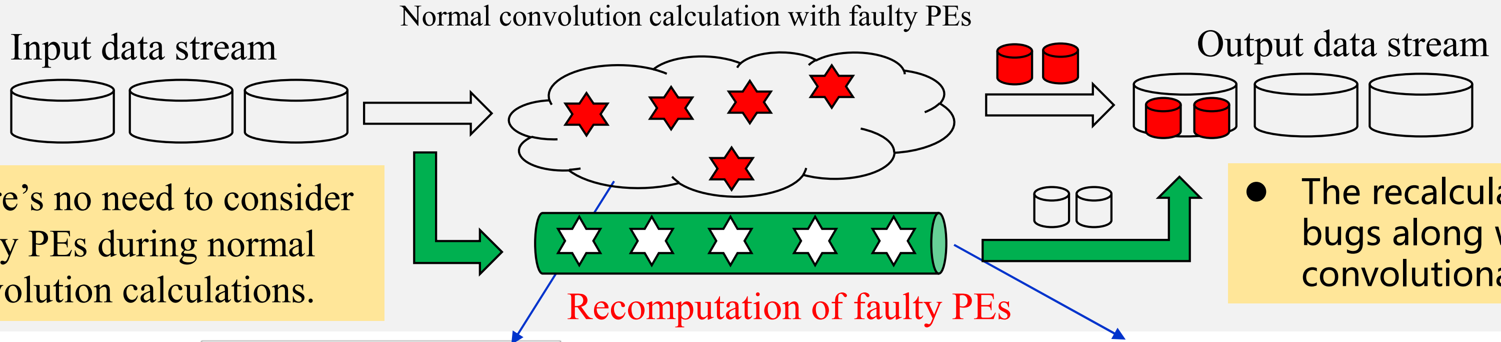
Fault distribution pattern 1



Fault distribution pattern 2

- Each redundant PE recovers any faulty PE in a limited region of the computing array while the region can be a row, a column, or both row and column.
- Be sensitive to the distribution of faults, and cannot tolerate various fault distributions.

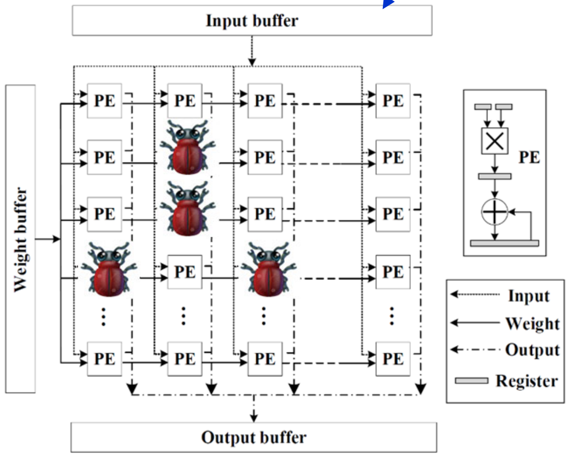
Recomputing based BIFT for Deep Learning Systems



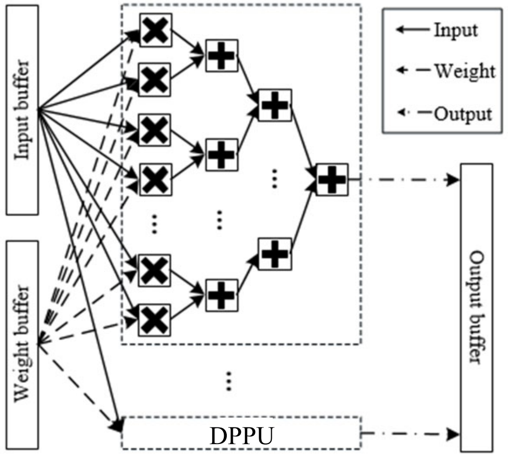
- There’s no need to consider faulty PEs during normal convolution calculations.

- The recalculation logic fixes bugs along with regular convolutional computations.

Each PE computes different neurons separately and serially.



Multiple PEs compute the same neuron in parallel.

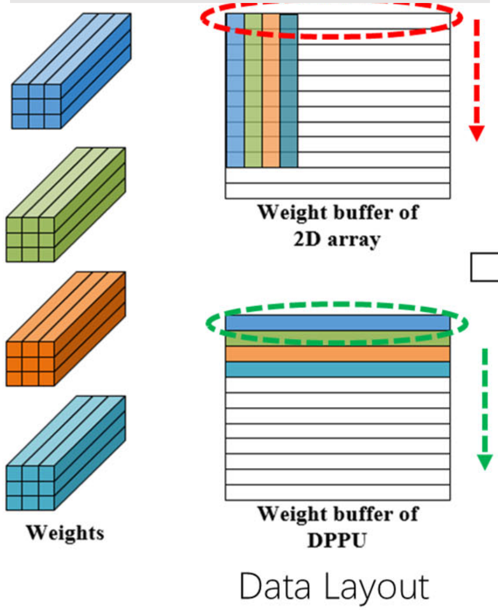


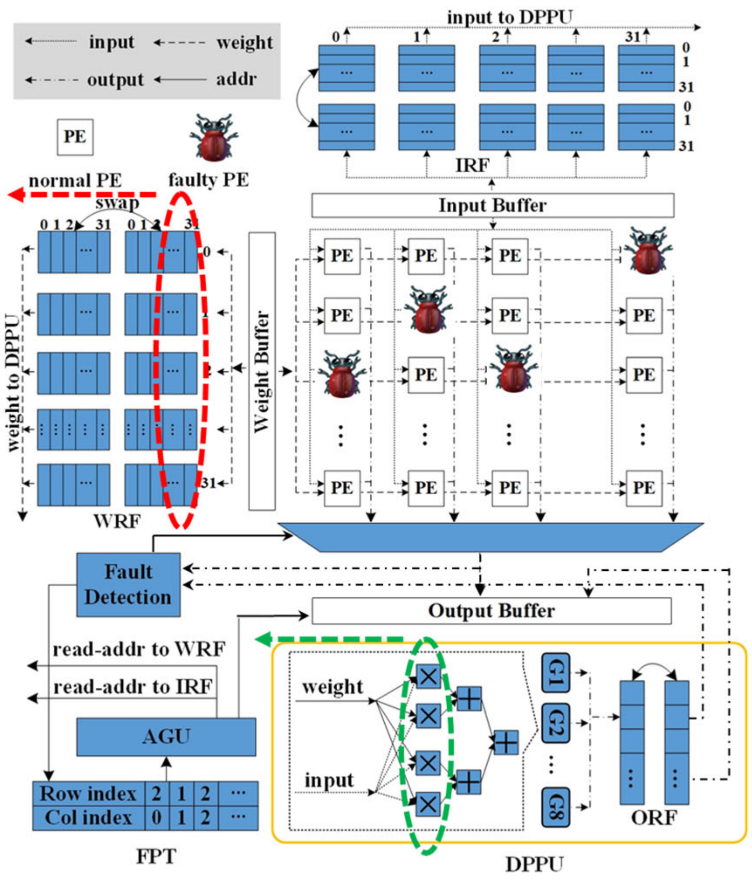
- HyCA: recomputation to continuously fix the erroneous neuron computation.
- Error repair with the help of parallel computation mode, FT completely transparent to software.

Recomputing based BIFT for Deep Learning Systems (HyCA)

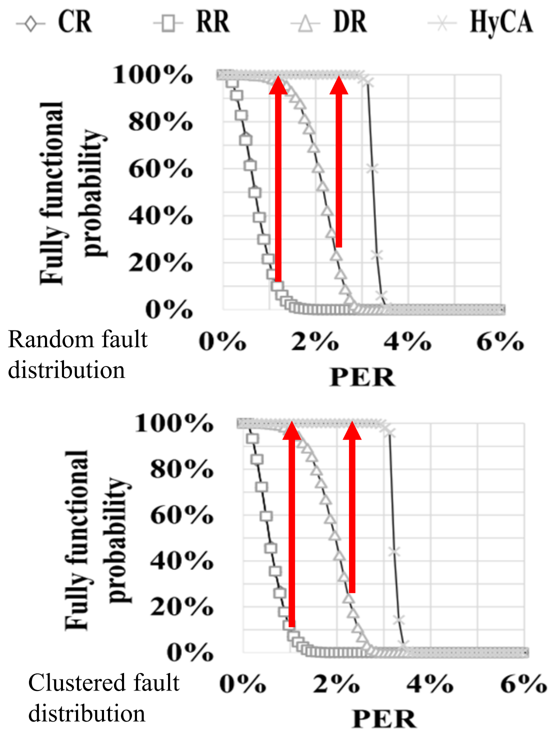
The DPPU can be used to repair faulty PEs at arbitrary locations.

reuses the cache data according to the location of the faulty PE





Significantly improves the success rate of complete repair



Cheng Liu, et al.: HyCA: A Hybrid Computing Architecture for Fault Tolerant Deep Learning (TCAD 2022)

Outline

1) SDCs on Computing Chips
2) Origins of SDCs
3) Testing against SDCs
4) Build-in fault-tolerance (BIFT) against SDCs
   a. BIFT for Multi-Core Systems
   b. BIFT for Deep Learning Systems
5) Concluding Thoughts

Concluding Thoughts on Testing

- For large-scale data center applications, **SDCs** caused by **circuit marginalities** that fail only under the specific conditions, can no longer be ignored.
- Improving test coverage of hardware is the fundamental reliability pursuit, while the cost of testing is high.
- Manufacturing testing against SDCs
  - targeting **statistical long paths** with the consideration of **path correlation** during delay testing to improve capture probabilities on **small delay defects**.
  - activating the **worst-case crosstalk/power noises** during test generation with the consideration of **layout** information
- System-level testing against SDCs
  - Test programs can be used to check every instruction, with **pseudo-random data** and **combinations of instructions**, and **repeated looping for millions of clock cycles,** to span the vast data, address, and instruction space

39

Concluding Thoughts on BIFT

- It is important to have **light-weight error detection** mechanisms with **low error detection latency** to guarantee real-time recovery in critical applications like in the automotive scenario.
- For the large-scale PEs in deep learning accelerators, it is better to have a **build-in architecture-level fault tolerance** to repair the faults of arbitrary distributions, such as the presented HyCA.
- More architecture-level fault tolerance solutions can be designed for different computing architectures.
- **Silicon lifecycle management**, with **sensor-rich architecture** will become a promising system-level solution.
  - PVT monitors, path margin monitors, functional monitors
  - DFT & BIST resources reused at system level
  - Data-driven learning-based approaches using chip telemetry infrastructure

Thanks & Q A

Thanks to my colleagues and students:

Minjin Zhang, Xiang Fu, Zijian He, Dawen Xu, etc. for work on delay testing of SDFs,

Tiancheng Wang, Yonghao Wang, etc., for work on instruction-level testing,

Cheng Liu, Zhefan Lv, etc. for work on build-in fault tolerance.

中科鉴芯 CASTEST http://www.castest.com.cn/

Providing Silicon Life-Cycle Solutions for Test & Reliability

TestConX中国 China Shanghai ▪ October 31, 2024

41

COPYRIGHT NOTICE